22

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LINCOLN LABORATORY

TX-2 TECHNICAL MANUAL

LINCOLN MANUAL NO. 44

Volume 3

JULY 1961

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology, with the joint support of the U.S. Army, Navy and Air Force under Air Force Contract AF 19(604)-7400.

LEXINGTON MASSACHUSETTS

TX-2 TECHNICAL MANUAL

TABLE OF CONTENTS

VOLUME I

VOLUME II

VOLUME III

CHAPTER 16

TIMING CHARTS

TABLE OF CONTENTS

CHAPTER 16

TIMING CHARTS

16-1 GENERAL INTRODUCTION

This chapter is a compilation of all the computer timing charts. The pulse and level notation used on these charts is described in Chapter 8. The timing charts vary in format and content, but generally they are arranged to show the events initiated by the various counter time levels. To facilitate their use, the charts have been arranged in the following groupings:

START-STOP CYCLE

- ADK (Alarm Delay Counter)
- SSC (Start-Stop Control)

SPECIAL CONTROL CYCLES

- CSK (Change of Sequence Counter)
- FK (F Memory Counter)
- XWK (X Memory Write Counter)

PK-QK MEMORY CYCLES

- PK (PKM^S, PKM^T, PKM^U, PKM_{VFF} and $PKM_{\overline{VFF}}$)
- QK (QKM^S, QKM^T, QKM^U, QKM_{VFF} and $QKM_{\overline{VFF}}$)

PK-QK INSTRUCTION CYCLES

OPR^{IOS}	EXX	STE	JOV
OPR^{AE}	ADX	FLF	JPA
JMP	DPX	FLG	JNA
JPX	SKM	ST-	EXA
JNX	LDE	ITE	INS
AUX	SPF	ITA	COM
RSX	SPG	UNA	TSD
SKX	LD-	SED	

PK-QK-AK INSTRUCTION CYCLES

CY-, SC-	MUL	DSA	ADD, SUB
DIV	NO-	TLY	

Generally it is necessary to examine one or more charts from each of the above groups in order to see the overall activity taking place in the execution of an instruction. For example, suppose a LDA is executed. First the PK Memory Timing Chart is examined to determine the events taking place during the instruction memory cycle. Since XWK is started during the PK memory cycle, the XWK timing chart is also examined. Next the QK Memory Timing Chart is examined to determine the events taking place during the operand

memory cycle. Since FK is started during the QK Memory cycle, the FK timing chart is also investigated. The operand instruction logic (as distinct from the operand memory logic) is found on the LDA Instruction Timing Chart. This too is examined. In this way a composite picture of the activity taking place during a LDA is obtained.

A brief description of the significance of the logic found on the timing charts accompanies each timing chart. In the case of the Instruction Timing Charts, the description has been supplemented with diagrams showing the significant data transfers, logic nets, etc. Some of the Instruction Timing Charts have also been illustrated with specific numerical examples. The intent is to bring out the general and special features of each counter and OP code.

16-2 START-STOP CYCLE

16-2.1 INTRODUCTION

There are two closely integrated systems that influence the starting and stopping of the computer. These are the start-stop control system and the alarm processing system.

The start-stop control system is basically a complex synchronizer for the start-stop console pushbuttons. The heart of the alarm processing system is the ADK counter. ADK time levels are also used in the start-stop control system.

These two systems individually and jointly generate control levels which become inputs to the Control Element. The Control Element then directly controls the starting and stopping of the computer.

16-2.2 START STOP CONTROL

The computer can operate in any one of three push button modes:

1) Low speed repeat (LSR^1)
2) Low speed push button ($LSPB^1$)
3) Hi speed ($LSR^0 \cdot LSPB^0$ = hi speed)

The effect of depressing the START button depends on which of the three push button modes the computer is operating in and the condition of the alarm system. However, the effect of depressing the STOP button is always the same, i.e., it is independent of the push button mode.

Start. Pressing the START button sets the $START_1$ flip-flop in the START synchronizer and the $STOP_1$ flip-flop in the STOP synchronizer. The next alpha (α) pulse, after these flip-flops are set, sets the $START_2$ flip-flop to ONE. $START_2^1$ enters as a factor in all the interlock start conditions, i.e., in the $PI^{START}1$, $PI^{START}2$, QI^{START} and CSI^{START} levels. The following alpha pulse (the pulse after the one which set $START_2$) clears the $STOP_2$ flip-flop if the computer is not in the low speed repeat mode, i.e., if LSR^0. If the computer is in the low speed repeat mode (LSR^1), then the $STOP_2$ flip-flop is cleared by LSO. $STOP_2^0$ clears all the stop flip-flops, i.e., PKS_1 PKS_2, QKS and CSKS. These stop flip-flops enter into the interlock start conditions, i.e., PKS_1^0 is a factor in the $PI^{START}1$ level, etc. PKS_1^0, PKS_2^0, QKS^0, $CSKS^0$ and $START_2^1$ represent essentially all the control levels going to the Control Element from the start stop system.

Note that when $START_2^0$, the STOP IO Unit level is generated. This level stops all free-running IO Units.

Stop. Pressing the stop button clears the $START_1$ flip-flop. ($START_1$ is also cleared by the occurrence of a SYNC alarm or an AL level when the AUTO START switch is turned on). The $START_2$ flip-flop is cleared by either $START_1^0$ or AL. $START_2^0$ immediately turns off all the interlock start levels.

If the computer is in either of the low speed modes, the $STOP_2$ flip-flop is set 0.4 microsecond after it is cleared. When $STOP_2^1$, the stop switches on the console, i.e., STOP ON CSK, STOP ON QK, etc., are used to set the stop flip-flops, i.e., PKS_1, PKS_2, etc. In this way, only the interlock start levels selected by the console switches are turned off. The computer now stops only when it needs one of the selected levels in order to proceed.

If the computer is in the low speed push button mode (LSPB), the START button must be depressed in order for the computer to proceed, because in this case the START button clears $STOP_2$.

16-2.3 ADK (ALARM DELAY COUNTER)

ADK controls the alarm processing system. One of its major functions is to convert asynchronous inputs into synchronous alarm control levels which can be used by the central computer.

ADK is a modified two stage Gray code counter. It is modified in the sense that two delay units (ALD_1 and ALD_2) are an integral part of the counter's logical circuitry.

The counter starts only when an unsuppressed alarm occurs or when the SYNC SYSTEM STOP level is generated. Such an occurrence, as indicated by the presence of the AL level, triggers the ALD_1 delay unit. ALD_1^1 in turn places the counter in the 01 state. The counter stays in this state until ALD_1 times out synchronously. During this period a level is generated for the CHIME ON UNSUPPRESSED ALARMS.

At the end of the period (ALD_1^0), the counter goes into state 11 and ALD_2 is triggered. During this delay a preset level for the Control Element ($\xrightarrow{\text{PRESET}}$ CE) is generated if the AUTO START and PASOFA (Preset And Start Over After Alarm) switches are on. The flag in sequence 00 is also raised if the PASOFA switch is on.

After the delay is over (ALD_2^0), the counter will remain in state 11 unless the AUTO START switch is turned on or until the CLEAR UNSUPPRESSED ALARMS push button is depressed (CA_1^1). From state 11 the counter proceeds to state 10 at which time all the unsuppressed alarms are cleared.

Pressing the START button has no effect unless ADK is in state 00. If an unsuppressed alarm has occurred and the AUTO START switch is <u>not</u> on, then ADK remains in state 03 until the CLEAR UNSUPPRESSED ALARM button is pushed. Note that the CALACO button first clears the unsuppressed alarms and then generates a START pulse.

ADK: ALARM DELAY COUNTER

ADK_2	ALD_2	ADK_1	ALD_1	TIME		
0	0	0	0	NOTE #1	PB Start → ⇒ L1→ $STOP_1$, L1→ $START_1$ $START_2^0$ · $START_1'$ ⇒ L1→ $START_2$ AL ⇒ L1→ ALD_1	
0	0	0	1	.4μs	ALD_1' ⇒ L1→ ADK_1	
0	0	1	1	7ms	ALD_1^{so} ⇒ L0→ ALD_1	NO CHIME ON $ALARMS_{usup}$ ⇒ CHIME ON $ALARMS_{usup}$
0	0	1	0	.4μs	ALD_1^0 ⇒ L1→ ADK_2, L1→ ALD_2	PB Clear $Alarms_{\mu sup}$ → ⇒ L1→ CUA
1	1	1	0	55ms	PASOFA + PB Startover ⇒ L1→ SYN_1 (STARTOVER SEQ.) PASOFA · AUTO-START ⇒ Preset → CE ALD_2^{so} ⇒ L0→ ALD_2	PB Clear $Alarms_{\mu sup}$ → ⇒ L1→ CUA
1	0	1	0	NOTE #2	CUA' + AUTO-START ⇒ L0→ ADK_1	
1	0	0	0	.4μs	$\overline{OCSAL_{sup}}$ ⇒ L0→ OCSAL $\overline{PSAL_{sup}}$ ⇒ L0→ PSAL $\overline{QSAL_{sup}}$ ⇒ L0→ QSAL $\overline{MPAL_{sup}}$ ⇒ L0→ MPAL $\overline{NPAL_{sup}}$ ⇒ L0→ NPAL $\overline{XPAL_{sup}}$ ⇒ L0→ XPAL $\overline{FPAL_{sup}}$ ⇒ L0→ FPAL $\overline{IOSAL_{sup}}$ ⇒ L0→ IOSAL $\overline{MISAL_{sup}}$ ⇒ L0→ MISAL L0→ ADK_2, L0→ CUA	
0	0	0	0	NOTE #1		

NOTES:

1. ADK NORMALLY SITS IN STATE 0000. THE INDEXING OF ADK IS INITIATED BY THE AL SIGNAL.

2. THIS TIME STATE IS .4μs IF THE AUTO-START SWITCH IS ACTIVATED OTHERWISE THE TIME INTERVAL DEPENDS ON THE MANUAL SETTING OF CUA VIA THE CLEAR $ALARMS_{\mu sup}$ SIGNAL

OCSAL' · $\overline{OCSAL_{sup}}$ ⇒ AL
PSAL' · $\overline{PSAL_{sup}}$ ⇒ AL
QSAL' · $\overline{QSAL_{sup}}$ ⇒ AL
MPAL' · $\overline{MPAL_{sup}}$ ⇒ AL
NPAL' · $\overline{NPAL_{sup}}$ ⇒ AL
XPAL' · $\overline{XPAL_{sup}}$ ⇒ AL
FPAL' · $\overline{FPAL_{sup}}$ ⇒ AL
IOSAL' · $\overline{IOSAL_{sup}}$ ⇒ AL
MISAL' · $\overline{MISAL_{sup}}$ ⇒ AL
MOUSETRAP' ⇒ AL

PB Clear $Alarms_{sup}$ → ⇒ L1→ CSAD (CSAD = CLEAR SUPPRESSED ALARMS DELAY: DE. = 100μs)

CSAD' · $OCSAL_{sup}$ ⇒ L0→ OCSAL
CSAD' · $PSAL_{sup}$ ⇒ L0→ PSAL
CSAD' · $QSAL_{sup}$ ⇒ L0→ QSAL
CSAD' · $MPAL_{sup}$ ⇒ L0→ MPAL
CSAD' · $NPAL_{sup}$ ⇒ L0→ NPAL
CSAD' · $XPAL_{sup}$ ⇒ L0→ XPAL
CSAD' · $FPAL_{sup}$ ⇒ L0→ FPAL
CSAD' · $IOSAL_{sup}$ ⇒ L0→ IOSAL
CSAD' · $MISAL_{sup}$ ⇒ L0→ MISAL
PB Preset → ⇒ L0→ TSAL, L0→ USAL, L0→ MOUSETRAP
PB Clear $Alarms_{usup}$ → ⇒ L0→ SYAL

SYD' ⇒ CHIME ON $ALARMS_{usup}$

AL = ALARM
CAU = CLEAR UNSUPPRESSED ALARMS
PASOFA = PRESET AND STARTOVER AFTER ALARM
CALACO = PB Clear $Alarms_{sup}$ → · PB Clear $Alarms_{usup}$ → · PB Start →
CODABO = PB Stop → · PB Clear $Alarms_{sup}$ → · PB Clear $Alarms_{usup}$ → · PB Preset → · PB Startover · PB Start →
$START_2^0$ = STOP UNIT

C. A. NORMAN 3-6-61

16-3 SPECIAL CONTROL CYCLES

16-3.1 INTRODUCTION

The timing charts in this section cover the events initiated by three special purpose counters. These counters are CSK, FK and XWK.

The FK and XWK counters control the F and X Memory systems, respectively. FK controls the read-write process in the F Memory, while XWK controls the write process in the X Memory (the read process is controlled by PK and CSK time levels). By having these processes controlled by independent counters, it is possible to initiate the processes at several different PK, QK and CSK times. Normally the XWK counter is started in $PK^{14\alpha}$ and FK in $QK^{00\alpha}$.

The CSK counter has a double function. It controls the events that occur during a change of sequence, and is also used as a delay synchronization counter in the PK waiting states.

16-3.2 CSK (CHANGE OF SEQUENCE COUNTER)

CSK is a modified four stage counter. It counts in states 00 through 07, when CSK_4^0 and in states 08 through 11, when CSK_4^1. In the first instance, CSK is interpreted as the change of sequence counter (CSK); and in the second instance, as the delay synchronization counter (DSK).

CSK cannot start unless the CSI^{START} condition is generated. On the other hand, DSK cannot count (assuming CSK_4^1 and PK is in one of the waiting states, i.e., PK^{00}, PK^{02} or PK^{23}) unless XWK is in its 00 resting state.

Change of Sequence. When CSK starts it transfers the address of the next program counter from the output of the J Coder into N_j. Two possibilities exist:

1) If the selected register is number 00, then a ZERO is placed in X and the contents of TSP is placed in $N_{2,1}$. XAS is cleared in this case so that the X Adder (XA) contains the content of $N_{2,1}$, i.e., TSP.

2) If the selected register is not register number 00, XAS is set and the content of the selected X Memory register is then read into X. Since $N_{2,1}$ was previously cleared, the sum formed in the X Adder is just the content of X.

In either case, the X Adder contains the value of the new program counter. The contents of K and N_j, representing the numbers of the old and new program counters, respectively, are saved in E. In CSK^{03}, the content of K and N_j are interchanged. The flag of the Trapping Sequence is also raised at this time if the 2.9 bit of the new program counter is a ONE and the mode of the Trapping Sequence asks for this information. In $CSK^{04\alpha}$, the value of the new program counter is copied into P, while simultaneously the value of the old program counter is copied into both X and $E_{2,1}$.

The value of the old program counter (now in X) is stored in the X Memory register specified by N_j (N_j now contains the address of the old program counter) by starting the XWK counter in $CSK^{04\alpha}$. X is also cleared at this time.

PI_3 is cleared in $CSK^{04\alpha}$, and in $CSK^{05\alpha}$ the flag of sequence number 00 is cleared if the new sequence is sequence number 00.

Finally in $CSK^{07\alpha}$ a certain amount of logic is performed which takes further into account the requirements of the Trapping Sequence. The information in E is placed in M. If a change to the Trapping Sequence has just occurred, because of a trap on the 2.9 bit of a program counter, the content of M is simultaneously placed in E. The content of M represents information left over from the previous CSK cycle. If the CSK cycle is one in which a trap on the 2.9 bit of the new program counter occurs, then this is indicated in $CSK^{07\alpha}$ by the $SS^{CH\ REQ}$ level and causes PI_3 to be set. The fact that PI_3 is set causes

the current CSK cycle to be followed immediately by another CSK cycle.

Delay Synchronization. CSK_4 is set whenever the computer is to wait in "limbo" for some interlock condition to change. In this case DSK simply counts from 08 through 11 repetitively. Finally the interlock change will occur and DSK^{11} will sample the desired interlock conditions. At this time CSK_4 is cleared and the counter goes into $CSK^{00\alpha}$.

Each time DSK enters state 11, an IOI clock pulse is fired off (except when either CSK_4 is being cleared or the QK cycle of a TSD is being performed).

In the PK^{00} waiting state, DSK cycles until the flag of a sequence (of any priority) goes up. In the $PK^{02\alpha}$ waiting state, DSK cycles until either the Arithmetic Element prediction net (AEI) indicates that the Arithmetic Element will soon be available or, if the previous instruction did not hold, until a sequence with a higher priority wants attention ($PI^{AE\ CH\ SEQ}$).

In the $PK^{23\alpha}$ waiting state, DSK cycles either until the $PI^{\overline{WAIT}}$ level indicates the current instruction can proceed or until some other sequence requests attention via the $PI^{LEAVE\ SEQ}$ level. Whether this other sequence is a sequence of any priority or one of a higher priority depends on whether the current instruction is a TSD or whether it is an instruction which does not hold, respectively.

CSK: CHANGE SEQUENCE COUNTER

	00	α	CSI^{start} · · · · · ⊃ $\overline{CSI^{start}}$ · · · · · ⊃	$JC \xrightarrow{\;\mid\;} N_{3.6-3.1}$ $\overline{CSK'} + 1 \nrightarrow CSK$
	01	α		$\lfloor 1 \rightarrow XR$, $\lfloor 1 \rightarrow XB$, $\lfloor 1 \rightarrow XAC$ $\lfloor 0 \rightarrow E$ $\lfloor 0 \rightarrow N_{2,1}$
	02	α	XR' (PLUS .06 μs DE.) · ⊃ N_J^{00} · · · · · · · ⊃ $N_J^{00} + (K^{eqJ} \cdot XPS')$ ⊃ $\overline{N_J^{00}}$ · · · · · · · ⊃ $\overline{N_J^{00}} \cdot (\overline{K^{eqJ}} + XPS^{\circ})$ ⊃	$\lfloor 0 \rightarrow XR$ $TSP \xrightarrow{1} N_{2,1}$, $\lfloor 0 \rightarrow XAS$ $\lfloor 0 \rightarrow X$, $\lfloor 1 \rightarrow X_P$ $\lfloor 1 \rightarrow XAS$ $XM \xrightarrow{\;\mid\;} X, X_P$ $K \xrightarrow{1} E_{4.6-4.1}$ $N_{3.6-3.1} \xrightarrow{1} E_{3.6-3.1}$
	03	α	 $\overline{\text{TRAP SEQUENCE}_{\text{COMPUTER SYNC}}} \cdot TP' \cdot \overline{KD^{42}} \cdot X'_{2.9}$ ⊃	$K \xrightarrow{\;\mid\;} N_{3.6-3.1}$ $N_{3.6-3.1} \xrightarrow{\;\mid\;} K$ $\lfloor \text{raise} \rightarrow$ Flag $^{42(\alpha)}$ (TRAP SEQ.)
CSK_4^{0}	04	α	 $XPAL_{sup} + XPAL^{\circ}$ · · ⊃ XP_{19}^{ev} · · · · · ⊃	$\lfloor 1 \rightarrow XR$, $\lfloor 1 \rightarrow XB$, $\lfloor 1 \rightarrow XPS$, $\lfloor \text{start} \rightarrow XWK$ $\lfloor 0 \rightarrow PI_3$ $XA \xrightarrow{\;\mid\;} P$ $P \xrightarrow{\;\mid\;} E_{2,1}$ $P \xrightarrow{\;\mid\;} X$ (NOTE XPAL IS ALWAYS IS ALWAY "O" AT THIS TIME) $\lfloor 1 \rightarrow XPAL$
	05	α	XR' (PLUS .06 μs DE.) · ⊃ KD^{00} · · · · · · ⊃	$\lfloor 0 \rightarrow XR$ $\lfloor \text{dismiss}_{KD} \rightarrow$ FLAG
	06	α	 $SS^{ch\ req}$ · · · · ⊃ $\overline{\text{TRAP SEQUENCE}_{\text{COMPUTER SYNC}}} \cdot TP' \cdot \overline{KD^{42}} \cdot X'_{2.9}$ ⊃	$E \xrightarrow{0,1} M$ $\lfloor 0 \rightarrow PI_2$, $\lfloor 0 \rightarrow PI_5$, $\lfloor 1 \rightarrow PI_3$ $M \xrightarrow{0,1} E$
	07	α		($\lfloor 00 \rightarrow CSK$)

DSK: DELAY SYNCHRONIZER COUNTER

	08	α	SEE NOTES 1 & 2	
	09	α	SEE NOTES 1 & 2	
	10	α	SEE NOTES 1 & 2	

NOTE 1: $CSK_4^{1} \cdot (\overline{PK^{00\alpha}} \cdot \overline{PK^{02\alpha}} \cdot \overline{PK^{23\alpha}} + \overline{XWK^{00}}) \supset \overline{CSK'} + 1 \nrightarrow CSK$

NOTE 2: $CSK_4^{1} \cdot K_{3.6}^{1} \supset KD \neq K$

$SS^{att\ req}$ = AT LEAST ONE FLAG IS UP OTHER THAN $\overline{KD]}$. (SEE NOTE 2)

$SS^{ch\ req}$ = A FLAG OF HIGHER PRIORITY THAN $\overline{KD]}$ IS UP. (SEE NOTE 2)

CSK_4^{1}	11	α	SEE NOTES 1 & 2 $\overline{\lfloor 0 \rightarrow CSK_4}$ · · · · · · · · · · ⊃ $\overline{\lfloor 0 \rightarrow CSK_4} \cdot (\overline{QKIR^{tsd}} + QB^{\circ})$ · · ⊃ $PK^{00\alpha} \cdot SS^{att\ req} \cdot (\overline{KD^{eqJC}} + KD^{00})$ ⊃ $PK^{00\alpha} \cdot SS^{att\ req}$ · · · · · · · ⊃ $PK^{02\alpha} \cdot \overline{AEI}$ · · · · · · · · ⊃ $PK^{02\alpha} \cdot PI^{AE\ ch\ seq}$ · · · · · · · ⊃ $PK^{23\alpha} \cdot \overline{PI^{wait}}$ · · · · · · · ⊃ $PK^{23\alpha} \cdot \overline{PI^{wait}} \cdot PKIR^{QK}$ · · · ⊃ $PK^{23\alpha} \cdot PI^{leave\ seq}$ · · · · · · ⊃ $PK^{23\alpha} \cdot PI^{leave\ seq} \cdot PKIR^{XM}$ · · ⊃	 $\lfloor 08 \rightarrow CSK$ IOI CLOCK PULSE $\lfloor 1 \rightarrow PI_3$ $\lfloor 0 \rightarrow CSK_4$ $\lfloor 0 \rightarrow CSK_4$ $\lfloor 0 \rightarrow CSK_4$, $\lfloor 00 \rightarrow PK$, $\lfloor 0 \rightarrow PI_2$, $\lfloor 1 \rightarrow PI_3$, $\lfloor 0 \rightarrow PI_5$ $\lfloor 0 \rightarrow CSK_4$, $\lfloor 24 \rightarrow PK$ $\lfloor 1 \rightarrow PI_1$ $\lfloor 0 \rightarrow CSK_4$, $\lfloor 00 \rightarrow PK$, $\lfloor 1 \rightarrow PI_3$ $\lfloor \text{start} \rightarrow XWK$

$$LIMBO = CSK_4^{1} \cdot \overline{[CSK_4^{1} \cdot (\overline{PK^{00\alpha}} \cdot \overline{PK^{02\alpha}} \cdot \overline{PK^{23\alpha}} \cdot \overline{XWK^{00}})]} = CSK_4^{1} \cdot XWK^{00} \cdot (PK^{00\alpha} + PK^{02\alpha} + PK^{23\alpha})$$

$$CSI^{start} = PK^{00\alpha} \cdot PI_1^{\circ} \cdot PI_3^{1} \cdot XW^{\circ} \cdot XB^{\circ} \cdot EB^{\circ} \cdot CSKS^{\circ} \cdot START_2^{1}$$

$$PI^{AE\ ch\ seq} = AEI \cdot SS^{att\ req} \cdot PI_4^{\circ}$$

$$PI^{leave\ seq} = PI^{AE\ ch\ seq} \cdot [PKIR^{AE} + (PKIR^{QK} \cdot XA^{AE})] + SS^{att\ req} \cdot PKIR^{tsd} \cdot [IOCM^{BB} + (QKIR^{tsd} \cdot QB')]$$

$$PI^{wait} = AEI \cdot [PKIR^{AE} + (PKIR^{QK} \cdot XA^{AE})] + PKIR^{tsd} \cdot [IOCM^{BB} + (QKIR^{tsd} \cdot QB')]$$

C A NORMAN 5-26-61

16-3.3 FK (F MEMORY COUNTER)

FK is used, in conjunction with a pulse delay line, to control the load and store processes involving the thin magnetic film F Memory. In these processes FK time levels (either directly or via a delay line) are used to control the E and $QKIR_{CF}$ registers and the F Memory read-write process.

FK is used in this manner for two purposes:

1) To read a configuration word into $QKIR_{CF}$ from the F Memory register. In this case FK runs from FK^{00} to FK^{02}. This occurs in all the operand type instructions (except the F Memory instructions themselves). In these instructions FK runs during the QK cycle. FK is also used to read out a configuration word during JPA, JNA and JOV. These are non-operand type instructions and FK runs in them during the PK cycle.

2) To execute the F Memory instructions themselves. In the case of FLF and FLG, FK reads out the content of the specified F Memory register(s) into the E register. Conversely, in the case of SPF and SPG the specified F register(s) are loaded from E. The execution logic of these instructions require that FK permute the content of E. (See discussion of FLF, FLG, SPF and SPG Instruction Time Charts.)

STARTING CONDITIONS. During most instructions FK is started at $QK^{00\alpha}$ when QK starts. However, for the JA instructions and FLF and FLG, FK is started via the FI interlock and, in the case of SPF and SPG, FK is started in the QK cycle at $QK^{13\alpha}$. Note that, in the case of FLF, QK does not start until the QI^{START} condition is generated at $FK^{02\alpha}$ when FI is set. Similarly, in the case of FLG, the QI^{START} condition is not generated until $FK^{07\alpha}$.

DELAY LINE. FK initiates the F Memory read-write cycle by pulsing the F Memory delay lines in the even numbered FK states, excluding the terminal state of FK. Thus, when only one F Memory cycle is required, the delay line is pulsed only in $FK^{00\alpha}$, even though FK runs through states 00, 01 and 02. Similarly, is the case of SPG and FLG, the delay line is pulsed only four times, even though FK runs through states 00, 01, 02, 03, 04, 05, 06, 07 and 08.

FK COUNTER CONFIGURATION MEMORY

FK8	FK		Configuration	Time	Action
	0	α	$\lfloor start \rightarrow FK \supset$ PULSE DELAY LINE $\lfloor start \rightarrow FK \supset \overline{FK'} + 1 \rightarrow FK$ $FC^{0} \supset \lfloor 0 \rightarrow FW$ $FC^{1} \supset \lfloor 1 \rightarrow FW$ } NOTE: FC ALWAYS ZERO	.12	
				.14	$FC^{0} + PKIR^{lf} \supset \lfloor 0 \rightarrow QKIR_{cf}$
	1	α		.26	$\lfloor 1 \rightarrow FR$
				.42	$FC^{0} \supset \lfloor 1 \rightarrow FW$; $\overline{PKIR^{lf}} \cdot \overline{PKIR^{00}_{cf}} \supset CFM \rightarrow QKIR_{cf}$
				.44	$PKIR^{lf} \cdot \overline{PKIR^{00}_{cf}} \supset E_{1} \xrightarrow{1} QKIR_{cf}$
				.54	$\lfloor 0 \rightarrow FR$; $PKIR^{00}_{cf} \supset \lfloor 0 \rightarrow QKIR_{cf}$
				.64	$FC^{0} \supset \lfloor 0 \rightarrow FW$
		β	$PKIR^{f} \supset E_{4} \rightarrow E_{3},\ E_{3} \rightarrow E_{2},\ E_{2} \rightarrow E_{1}$ $PKIR^{lf} \supset E_{1} \rightarrow E_{4}$ $PKIR^{sf} \supset QKIR_{cf} \rightarrow E_{4}$		
	2	α	$PKIR^{ff} \cdot (FP^{odd}_{10} + FPAL_{sup} + PKIR^{00}_{cf}) \supset$ PULSE DE LINE $\overline{PKIR^{00}_{cf}} \cdot FP^{ev}_{10} \supset \lfloor 1 \rightarrow FPAL$ $\overline{PKIR^{ff}} \supset \lfloor 0 \rightarrow FK,\ \lfloor 0 \rightarrow FK8,\ \overline{FK'} + 1 \rightarrow FK$ $\overline{PKIR^{ff}} \cdot PKIR^{sf} + PKIR^{ja} \supset \lfloor 1 \rightarrow FI$	.12	$\overline{PKIR_{cf}'} + 1 \rightarrow PKIR_{cf}$
				.14	$FC^{0} + PKIR^{lf} \supset \lfloor 0 \rightarrow QKIR_{cf}$
		β	$PKIR^{flf} \supset E_{4} \rightarrow E_{1},\ E_{3} \rightarrow E_{4},$ $E_{2} \rightarrow E_{3},\ E_{1} \rightarrow E_{2}$	.26	$\lfloor 1 \rightarrow FR$
				.42	$FC^{0} \supset \lfloor 1 \rightarrow FW$; $\overline{PKIR^{lf}} \cdot \overline{PKIR^{00}_{cf}} \supset CFM \rightarrow QKIR_{cf}$
	3	α		.44	$PKIR^{lf} \cdot \overline{PKIR^{00}_{cf}} \supset E_{1} \xrightarrow{1} QKIR_{cf}$
				.54	$\lfloor 0 \rightarrow FR$; $PKIR^{00}_{cf} \supset \lfloor 0 \rightarrow QKIR_{cf}$
				.64	$FC^{0} \supset \lfloor 0 \rightarrow FW$
		β	$PKIR^{f} \supset E_{4} \rightarrow E_{3},\ E_{3} \rightarrow E_{2},\ E_{2} \rightarrow E_{1}$ $PKIR^{lf} \supset E_{1} \rightarrow E_{4}$ $PKIR^{sf} \supset QKIR_{cf} \rightarrow E_{4}$		
$FK8^{0}$	4	α	$PKIR^{ff} \cdot (FP^{odd}_{10} + FPAL_{sup} + PKIR^{00}_{cf}) \supset$ PULSE DE LINE $\overline{PKIR^{00}_{cf}} \cdot FP^{ev}_{10} \supset \lfloor 1 \rightarrow FPAL$	.12	$\overline{PKIR_{cf}'} + 1 \rightarrow PKIR_{cf}$
				.14	$FC^{0} + PKIR^{lf} \supset \lfloor 0 \rightarrow QKIR_{cf}$
		β		.26	$\lfloor 1 \rightarrow FR$
				.42	$FC^{0} \supset \lfloor 1 \rightarrow FW$; $\overline{PKIR^{lf}} \cdot \overline{PKIR^{00}_{cf}} \supset CFM \rightarrow QKIR_{cf}$
	5	α		.44	$PKIR^{lf} \cdot \overline{PKIR^{00}_{cf}} \supset E_{1} \xrightarrow{1} QKIR_{cf}$
				.54	$\lfloor 0 \rightarrow FR$; $PKIR^{00}_{cf} \supset \lfloor 0 \rightarrow QKIR_{cf}$
				.64	$FC^{0} \supset \lfloor 0 \rightarrow FW$
		β	$PKIR^{f} \supset E_{4} \rightarrow E_{3},\ E_{3} \rightarrow E_{2},\ E_{2} \rightarrow E_{1}$ $PKIR^{lf} \supset E_{1} \rightarrow E_{4}$ $PKIR^{sf} \supset QKIR_{cf} \rightarrow E_{4}$		
	6	α	$PKIR^{ff} \cdot (FP^{odd}_{10} + FPAL_{sup} + PKIR^{00}_{cf}) \supset$ PULSE DE LINE $\overline{PKIR^{00}_{cf}} \cdot FP^{ev}_{10} \supset \lfloor 1 \rightarrow FPAL$ $QKIR^{spg} \supset \lfloor 0 \rightarrow EB$	.12	$\overline{PKIR_{cf}'} + 1 \rightarrow PKIR_{cf}$
				.14	$FC^{0} + PKIR^{lf} \supset \lfloor 0 \rightarrow QKIR_{cf}$
		β		.26	$\lfloor 1 \rightarrow FR$
				.42	$FC^{0} \supset \lfloor 1 \rightarrow FW$; $PKIR^{lf} \cdot \overline{PKIR^{00}_{cf}} \supset CFM \rightarrow QKIR_{cf}$
	7	α	$PKIR^{sf} \supset \lfloor 1 \rightarrow FI$ $\lfloor 1 \rightarrow FK8$	.44	$PKIR^{lf} \cdot \overline{PKIR^{00}_{cf}} \supset E_{1} \xrightarrow{1} QKIR_{cf}$
				.54	$\lfloor 0 \rightarrow FR$; $PKIR^{00}_{cf} \supset \lfloor 0 \rightarrow QKIR_{cf}$
				.64	$FC^{0} \supset \lfloor 0 \rightarrow FW$
		β	$PKIR^{f} \supset E_{4} \rightarrow E_{3},\ E_{3} \rightarrow E_{2},\ E_{2} \rightarrow E_{1}$ $PKIR^{lf} \supset E_{1} \rightarrow E_{4}$ $PKIR^{sf} \supset QKIR_{cf} \rightarrow E_{4}$		
$FK8^{1}$	0	α	$\overline{PKIR^{00}_{cf}} \cdot FP^{ev}_{10} \supset \lfloor 1 \rightarrow FPAL$ $\lfloor 0 \rightarrow FK8$		

$PKIR^{lff} = PKIR^{3X} \cdot PKIR^{X2}$

$PKIR^{lf} = PKIR^{spf} + PKIR^{spg}$

$PKIR^{sf} = PKIR^{flf} + PKIR^{flg}$

$PKIR^{ff} = PKIR^{spg} + PKIR^{flg}$

$PKIR^{f} = PKIR^{spf} + \overline{PKIR^{spg}} + PKIR^{flf} + PKIR^{flg}$

$\lfloor start \rightarrow FK = QK^{00d} \cdot QI^{start} \cdot \overline{PKIR^{f}} \cdot \overline{PKIR^{skm}} + QK^{13\alpha} \cdot (QKIR^{spg} + QKIR^{spf}) + FI^{0} \cdot EB^{0} \cdot (PKIR^{sf} + PKIR^{ja})$

C.A. NORMAN 2-20-61

16-3.4 XWK (X MEMORY WRITE COUNTER)

XWK

This counter controls the logic used in writing the content of the X register into the selected X Memory register. The counter does not control the logic used in reading out the contents of a selected X Memory register into X. This is accomplished by the PK or CSK counters.

XWK sets the XW interlock to ONE in $XWK^{02\alpha}$. This turns the WRITE current on. The XB interlock is cleared to ZERO at $XWK^{02\alpha}$ indicating that the X register will in 1.6 microseconds no longer be "busy" (XB^{0}).

The XW interlock is cleared to ZERO in $XWK^{06\alpha}$. This turns the WRITE current off and effectively ends the X write cycle.

The conditions for starting XWK are discussed in detail in Chapter 10. The interlocking of XWK with other counters is also discussed in Chapter 9.

XWK: X MEMORY WRITE COUNTER

00	α	$\overline{XWK'} + 1 \twoheadrightarrow XWK$ $\xrightarrow{\text{start}} XWK \supset \xrightarrow{L1} XWK$
01		
02	α	$\xrightarrow{L1} XW$ $\xrightarrow{L0} XB$
03		
04		
05		
06	α	$\xrightarrow{L0} XW$
07		

$CSK^{04\alpha} \cdots\cdots \supset \xrightarrow{\text{start}} XWK$

$PK^{14\alpha} \cdot (\overline{PKIR^{XM}} + PI^{1}_{2}) \cdots\cdots \supset \xrightarrow{\text{start}} XWK$

$PK^{23\alpha} \cdot PKIR^{XM} \cdot CSK^{11\alpha} \cdot PKIR^{\text{leave sequence}} \cdots \supset \xrightarrow{\text{start}} XWK$

$PK^{31\alpha} \cdot PKIR^{XM} \cdots\cdots \supset \xrightarrow{\text{start}} XWK$

$QK^{22\alpha} \cdot QKIR^{ld} \cdot QKIR^{X} \cdots\cdots \supset \xrightarrow{\text{start}} XWK$

$QK^{31\alpha} \cdot QKIR^{aux} \cdots\cdots \supset \xrightarrow{\text{start}} XWK$

$$PI^{AE\,ch\,seq} = AEI \cdot SS^{ch\,req} \cdot PI^{0}_{4}$$

$$PI^{leave\,seq} = PI^{AE\,ch\,seq} \cdot [PKIR^{AE} + (PKIR^{QK} \cdot XA^{AE})] + SS^{att\,req} \cdot PKIR^{tsd} \cdot [IOCM^{BB} + (QKIR^{tsd} \cdot QB^{1})]$$

$$PKIR^{jx} = PKIR^{0X}_{op} \cdot (PKIR^{X6}_{op} + PKIR^{X7}_{op}) = (JPX + JNX)$$

$$PKIR^{XM} = PKIR^{jx} + (PKIR^{jmp} \cdot PKIR_{cf^{1}_{2}}) = (JPX + JNX + {}^{XXX1X}JMP)$$

$$QKIR^{X} = (QKIR^{1X}_{op} \cdot QKIR^{X1}_{op}) + (QKIR^{1X}_{op} \cdot QKIR_{op^{1}_{3}} \cdot QKIR_{op^{0}_{1}}) = (RSX + EXX + DPX)$$

C. A. NORMAN 2-23-61

16-4 PK-QK MEMORY CYCLES

16-4.1 INTRODUCTION

This section covers the PK and QK Memory Timing Charts. All of the instruction, deferred-address and operand read-write control is found on these charts. The charts also cover much of the interlock control, waiting state logic, In-Out control, alarm control, X Memory control and a variety of miscellaneous control logic.

The PK and QK Memory Timing Charts have very similar formats. All of the memory dependent logic is columnated by memory, e.g., all the S Memory dependent logic is found under PKM^S or QKM^S, depending on whether an instruction or deferred address word, or an operand word, respectively, is involved. (It is worth noting the similarity of the entries in the corresponding PKM and QKM columns.) A miscellaneous column is included on each chart, and also one or more columns are included showing the basic interlock control.

The PK Memory timing chart covers PK^{00} through PK^{24}. The basic PKM cycle extends from PK^{00} through PK^{22}. PK^{23} and PK^{24} are special time levels used to determine what activity will follow the current PK cycle.

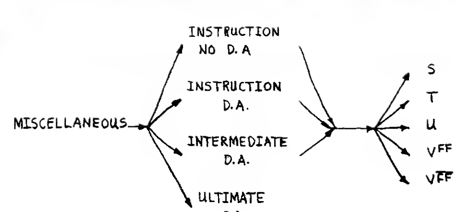
The QK Memory timing chart covers QK^{00} through QK^{31}. The basic QKM cycle extends from QK^{00} through QK^{31}. The actual states used depend on the specific instruction. In some cases the basic cycle is extended to accomodate the needs of the instruction logic.

Note that $V_{\overline{FF}}$ is usually referred to as a toggle switch memory, even though it also contains plugboard and other memory registers.

PK TIMING CHART

PK TIME LEVELS		INSTRUCTION WORD CYCLE (NO DEFERRED ADDRESSING)	MISCELLANEOUS CONTROLS	DEFERRED ADDRESS PK CYCLE: INSTRUCTION WORD CYCLE ($PI_2^0 \cdot N_{2.9}^1$)	DEFERRED ADDRESS PK CYCLE: INTERMEDIATE DEFERRED CYCLE ($PI_2^1 \cdot PI_5^1$)	DEFERRED ADDRESS PK CYCLE: ULTIMATE DEFERRED CYCLE ($PI_2^1 \cdot PI_5^0$)
00	α	$PI_2^0 \cdot \overline{PI^{start_1}} \Rightarrow \xrightarrow{1} PKA$ " " $\Rightarrow \xrightarrow{0} DFA$ DSK LOGIC $CSK_4^1 \cdot K_{34}^1 \Rightarrow KD \neq K$ $CSK^{11\alpha} \cdot SS^{att\,req} \cdot (K^{eqJC} + KD^{00}) \Rightarrow \xrightarrow{1} PI_3$ $CSK^{11\alpha} \cdot SS^{att\,req} \Rightarrow \xrightarrow{0} CSK_4$ $CSK^{11\alpha} \cdot \overline{\xrightarrow{0} CSK_4} \cdot (\overline{QKIR^{tsd}} + QB^0) \Rightarrow$ IOI CLOCK	$QK^{00\alpha} \Rightarrow SM^{off} \xrightarrow{j} SMOFF$ $QK^{00\alpha} \Rightarrow TM^{off} \xrightarrow{j} TMOFF$ $QK^{00\alpha} \Rightarrow UM^{off} \xrightarrow{j} UMOFF$	$PI_2^0 \cdot PI^{start_1} \Rightarrow \xrightarrow{1} PKA$ " " $\Rightarrow \xrightarrow{0} DFA$ DSK LOGIC $CSK_4^1 \cdot K_{34}^1 \Rightarrow KD \neq K$ $CSK^{11\alpha} \cdot SS^{att\,req} \cdot (K^{eqJC} + KD^{00}) \Rightarrow \xrightarrow{1} PI_3$ $CSK^{11\alpha} \cdot SS^{att\,req} \Rightarrow \xrightarrow{0} CSK_4$ $CSK^{11\alpha} \cdot \overline{\xrightarrow{0} CSK_4} \cdot (\overline{QKIR^{tsd}} + QB^0) \Rightarrow$ IOI CLOCK	$PI_2^1 \cdot \overline{PI^{start_2}} \Rightarrow \overline{PK}+1 \rightarrow PK$ $PI_5^1 \cdot PI^{start_2} \Rightarrow XA \xrightarrow{j} Q$ " " $\Rightarrow \xrightarrow{1} DFA$ $PI_5^1 \cdot \overline{PI^{start_2}} \Rightarrow \xrightarrow{0} PKA$ $PI_5^1 \cdot PI^{start_2} \Rightarrow \xrightarrow{1} PKA$	$PI_2^1 \cdot \overline{PI^{start_2}} \Rightarrow \overline{PK'}+1 \rightarrow PK$ $PI_2^1 \cdot PI^{start_2} \Rightarrow XA \xrightarrow{j} Q$ " " $\Rightarrow \xrightarrow{1} DFA$ $PI_2^1 \cdot PI_5^0 \Rightarrow \xrightarrow{0} PKA$
01	α		$\overline{QKIR^{tsd}} + QB^0 \Rightarrow$ IOI CLOCK		$PI_5^1 \cdot XAS^0 \Rightarrow \xrightarrow{0} QKIR_{cf}$	$PI_2^1 \cdot PI_5^0 \cdot \overline{MPA} \Rightarrow E \xrightarrow{0,1} M$ $PI_2^1 \cdot PI_5^0 \Rightarrow \xrightarrow{0} E$ $PKA^0 \Rightarrow \xrightarrow{09} PK$
02	α					
02	β					
03	β					
04	α					
05	α					
06	α					
09	α	$PI_2^0 \cdot PKM^{legal} \Rightarrow \xrightarrow{1} PSAL$		$PI_2^0 \cdot PKM^{legal} \Rightarrow \xrightarrow{1} PSAL$	$PI_5^1 \cdot PKM^{legal} \Rightarrow \xrightarrow{1} QSAL$ $PI_5^1 \cdot XAS^0 \Rightarrow N_j \xrightarrow{j} QKIR_{cf_{9-4}}$	$PI_2^1 \cdot PI_5^0 \Rightarrow QKIR_{cf_{9-4}} \xrightarrow{1} E_{3,6-3,1}$ " " $\Rightarrow XA \xrightarrow{1} E_{1,2}$
10	α		AUTO-START $\Rightarrow \xrightarrow{0} N_{4,2,1}$			$PI_2^1 \cdot PI_5^0 \Rightarrow \xrightarrow{0} N_{4,2,1}$
10	β					
11	α					$PI_2^1 \cdot PI_5^0 \Rightarrow M \xrightarrow{0,1} E$ " " $\Rightarrow E \xrightarrow{1,j} N$
11	β					
12	α	$PI_2^0 \Rightarrow N_{h,op} \xrightarrow{j} PKIR_{h,op}$	$\overline{QKIR^{tsd}} + QB^0 \Rightarrow$ IOI CLOCK $\xrightarrow{1} XB$ $\xrightarrow{1} XR$	$PI_2^0 \Rightarrow N_{h,op} \xrightarrow{j} PKIR_{h,op}$		
12	β					
13	α	$PI_2^0 \cdot N_{2.9}^0 \cdot (PKIR^{ja} + PKIR^{sf}) \Rightarrow \xrightarrow{0} FI$ $PI_2^0 \cdot N_{4,10}^1 \cdot TNI' \Rightarrow \xrightarrow{raise} FLAG_{42}$ $PI_2^0 \Rightarrow N_{cf} \xrightarrow{j} PKIR_{cf}$	XR^1 (PULS · 0.6μsec) $\Rightarrow \xrightarrow{0} XR$ $N_j^{00} \cdot (K^{eqJ} + XPS^0) \Rightarrow XM \xrightarrow{j} X$ $N_j^{00} + (K^{eqJ} \cdot XPS^1) \Rightarrow \xrightarrow{0} X$ " " " " $\Rightarrow \xrightarrow{1} X_P$	$PI_2^0 \cdot N_{2.9}^0 \cdot PKIR^{def} \Rightarrow \xrightarrow{1} PI_2$ $PI_2^0 \cdot N_{4,10}^1 \cdot TNI' \Rightarrow \xrightarrow{raise} FLAG_{42}$ $PI_2^0 \Rightarrow N_{cf} \xrightarrow{j} PKIR_{cf}$	$PI_2^1 \cdot PI_5^1 \cdot TNI' \Rightarrow \xrightarrow{raise} FLAG_{42}$	$PI_2^1 \cdot PI_5^0 \Rightarrow \xrightarrow{0} PI_2$ $PI_2^1 \cdot PI_5^0 \cdot (PKIR^{ja} + PKIR^{sf}) \Rightarrow \xrightarrow{0} FI$
13	β		FORCE XA CARRY			
14	α	$\overline{PKIR^{XM}} \Rightarrow \xrightarrow{start} XWK$ $PI_2^0 \cdot PKIR^{ind} \xrightarrow{j} XAS$		$PI_5 \xrightarrow{j} XAS$ $PI_2^1 \cdot N_{2.9}^1 \cdot PKIR^{def} \Rightarrow \xrightarrow{1} PI_5$ $PI_2^1 \Rightarrow \xrightarrow{start} XWK$	$PI_5 \xrightarrow{j} XAS$ $N_{2.9}^0 \cdot PI_2^1 \Rightarrow \xrightarrow{0} PI_5$ $PI_2^1 \Rightarrow \xrightarrow{start} XWK$	$\overline{PKIR^{XM}} \Rightarrow \xrightarrow{start} XWK$ $PI_2^0 \cdot PKIR^{ind} \xrightarrow{j} XAS$
15	α	$PI_2^0 \cdot PKIR_{op}^{0X} \cdot PKIR_{op}^{X6} \Rightarrow \xrightarrow{C} X$ $PI_2^0 \cdot PKIR^{skx} \cdot PKIR_{cf_{3,1}}^{0,1+1,0} \Rightarrow \xrightarrow{C} X$ $PI_2^0 \cdot \overline{PKIR^{def}} \Rightarrow \xrightarrow{1} OCSAL$	$K^{eqJ} \Rightarrow \xrightarrow{0} XPS$ $XP_{19}^{er} \Rightarrow \xrightarrow{1} XPAL$			$PI_2^0 \cdot [(PKIR_{op}^{0X} \cdot PKIR_{op}^{X6}) + (PKIR^{skx} \cdot PKIR_{cf_{3,1}}^{0,1+1,0})] \Rightarrow \xrightarrow{C} X$ $PI_2^0 \cdot \overline{PKIR^{def}} \Rightarrow \xrightarrow{1} OCSAL$ $PKA^0 \Rightarrow \xrightarrow{22} PK$
16	α					
22	α	$PI_2^0 \cdot PI^{leave\,seq.} \Rightarrow \xrightarrow{100} PK$ " " $\Rightarrow \xrightarrow{1} PI_3$ $PI_2^0 \cdot \overline{PI^{wait}} \Rightarrow \xrightarrow{24} PK$ $PI_2^0 \cdot (\overline{IOCM^{BB} + QB' \cdot QKIR^{tsd}}) \cdot PKIR^{tsd} \Rightarrow \xrightarrow[KD]{dismiss}$ FLAG $PI_2^0 \cdot \overline{PI^{wait}} \cdot PKIR^{QK} \Rightarrow \xrightarrow{1} PI_1$ $PI_2^0 \cdot PI^{wait} \cdot \overline{PI^{leave\,seq.}} \Rightarrow \xrightarrow{1} CSK_4$		$PI_2^1 \Rightarrow \xrightarrow{100} PK$	$PI_2^1 \Rightarrow \xrightarrow{100} PK$	$PI_2^0 \cdot PI^{leave\,seq.} \Rightarrow \xrightarrow{100} PK$ " " $\Rightarrow \xrightarrow{1} PI_3$ $PI_2^0 \cdot \overline{PI^{wait}} \Rightarrow \xrightarrow{24} PK$ $PI_2^0 \cdot (\overline{IOCM^{BB} + QB' \cdot QKIR^{tsd}}) \cdot PKIR^{tsd} \Rightarrow \xrightarrow[KD]{dismiss}$ FLAG $PI_2^0 \cdot \overline{PI^{wait}} \cdot PKIR^{QK} \Rightarrow \xrightarrow{1} PI_1$ $PI_2^0 \cdot PI^{wait} \cdot \overline{PI^{leave\,seq}} \Rightarrow \xrightarrow{1} CSK_4$
23	α	$\xrightarrow{0} PKA$ $\overline{PK}+1 \rightarrow PK$ DSK LOGIC $CSK^{11\alpha} \cdot \overline{PI^{wait}} \cdot PKIR^{QK} \Rightarrow \xrightarrow{1} PI_1$ $CSK^{11\alpha} \cdot \overline{PI^{wait}} \Rightarrow \xrightarrow{24} PK$ $CSK^{11\alpha} \cdot PI^{leave\,seq.} \Rightarrow \xrightarrow{100} PK$ " " $\Rightarrow \xrightarrow{1} PI_3$ $CSK^{11\alpha} \cdot (\overline{PI^{wait}} + PI^{leave\,seq.}) \Rightarrow \xrightarrow{0} CSK_4$ $CSK^{11\alpha} \cdot PI^{leave\,seq.} \cdot PKIR^{XM} \Rightarrow \xrightarrow{start} XWK$ $CSK^{11\alpha} \cdot \overline{\xrightarrow{0} CSK_4} \cdot (\overline{QKIR^{tsd}} + QB^0) \Rightarrow$ IOI CLOCK PK^{23} WAITING STATE				$\xrightarrow{0} PKA$ $\overline{PK'}+1 \rightarrow PK$ DSK LOGIC $CSK^{11\alpha} \cdot \overline{PI^{wait}} \cdot PKIR^{QK} \Rightarrow \xrightarrow{1} PI_1$ $CSK^{11\alpha} \cdot \overline{PI^{wait}} \Rightarrow \xrightarrow{24} PK$ $CSK^{11\alpha} \cdot PI^{leave\,seq} \Rightarrow \xrightarrow{100} PK$ " " $\Rightarrow \xrightarrow{1} PI_3$ $CSK^{11\alpha} \cdot (\overline{PI^{wait}} + PI^{leave\,seq}) \Rightarrow \xrightarrow{0} CSK_4$ $CSK^{11\alpha} \cdot PI^{leave\,seq.} \cdot PKIR^{XM} \Rightarrow \xrightarrow{start} XWK$ $CSK^{11\alpha} \cdot \overline{\xrightarrow{0} CSK_4} \cdot (\overline{QKIR^{tsd}} + QB^0) \Rightarrow$ IOI CLOCK PK^{23} WAITING STATE
24	α	$PKIR_h \xrightarrow{j} PI_4$ $\overline{P'}+1 \rightarrow P$ $\xrightarrow{0} PKA$ $\overline{PKIR^{dis}} \Rightarrow \xrightarrow{100} PK$ $\overline{PKIR^{ios}} \cdot PI^{ch\,seq.} \Rightarrow \xrightarrow{1} PI_3$				$PKIR_h \xrightarrow{j} PI_4$ $\overline{P'}+1 \rightarrow P$ $\xrightarrow{0} PKA$ $\overline{PKIR^{dis}} \Rightarrow \xrightarrow{100} PK$ $\overline{PKIR^{ios}} \cdot PI^{ch\,seq.} \Rightarrow \xrightarrow{1} PI_3$

PK TIME LEVELS		PK MEMORY CYCLES: S MEMORY CYCLE (PKM^S)	T MEMORY CYCLE (PKM^T)	U MEMORY CYCLE (PKM^U)	V^{FF} MEMORY CYCLE (PKM^{VFF})	$V^{\overline{FF}}$ MEMORY CYCLE ($PKM^{V\overline{FF}}$)	PK TIME LEVELS	
00	α	$PI_2^0 \cdot \overline{PI^{start_1}} \Rightarrow \overline{PK'}+1 \rightarrow PK$	$PI_2^0 \cdot \overline{PI^{start_1}} \Rightarrow \overline{PK}+1 \rightarrow PK$	$PI_2^0 \cdot \overline{PI^{start_1}} \Rightarrow \overline{PK}+1 \rightarrow PK$	$PI_2^0 \cdot \overline{PI^{start_1}} \Rightarrow \overline{PK'}+1 \rightarrow PK$	$PI_2^0 \cdot \overline{PI^{start_1}} \Rightarrow \overline{PK'}+1 \rightarrow PK$	α	00
01	α		$TMOFF^0 \Rightarrow$ PULSE TR DE. LINE $\xrightarrow{09} PK$	$UMOFF^0 \Rightarrow$ PULSE UR & DE. LINE $\xrightarrow{09} PK$			α	01
02	α				$VMD^{AE} \cdot AEI \cdot CSK_4^0 \Rightarrow \xrightarrow{1} CSK_4$ DSK LOGIC $CSK^{11\alpha} \cdot \overline{AEI} \Rightarrow \xrightarrow{0} CSK_4$ $CSK^{11\alpha} \cdot PI^{AE\,ch\,seq} \Rightarrow \xrightarrow{0} CSK_4$ " " $\Rightarrow \xrightarrow{0} PI_2, PI_5$ " " $\Rightarrow \xrightarrow{1} PI_3$ " " $\Rightarrow \xrightarrow{100} PK$ " " $\cdot \overline{\xrightarrow{0} CSK_4} \cdot (\overline{QKIR^{tsd}} + QB^0) \Rightarrow$ IOI CLOCK $CSK_4^0 \cdot EB^0 \cdot QB^0 \cdot (\overline{AEB} + \overline{VMD^{AE}}) \Rightarrow \xrightarrow{09} PK$ $\overline{PK'}+1 \rightarrow PK$ PK^{02} WAITING STATE	$\xrightarrow{09} PK$	α	02
02	β	$SMOFF^0 \Rightarrow \xrightarrow{1} SR_V$					β	
03	β	$SMOFF^0 \Rightarrow \xrightarrow{1} SR_u$					β	03
04	α						α	04
05	α						α	05
06	α	$\xrightarrow{09} PK$					α	06
09	α				$\overline{MPA} \Rightarrow E \xrightarrow{0,1} M$ $\overline{VMD^E} \Rightarrow \xrightarrow{0} E$		α	09
10	α	$PKM^{legal} \Rightarrow \xrightarrow{0} N_{4,2,1}$	$PKM^{legal} \Rightarrow \xrightarrow{0} N_{4,2,1}$	$PKM^{legal} \Rightarrow \xrightarrow{0} N_{4,2,1}$	$PKM^{legal} \Rightarrow \xrightarrow{0} N_{4,2,1}$ $\overline{VMD^E} \Rightarrow VFFM \xrightarrow{1} E$	$PKM^{legal} \Rightarrow \xrightarrow{0} N_{4,2,1}$	α	10
10	β	$SM \xrightarrow{1,j} N$					β	
11	α	$\xrightarrow{0} SR_V$			$E \xrightarrow{1,j} N$		α	11
11	β		$TM \xrightarrow{1,j} N$	$UM \xrightarrow{1,j} N$		$VTM \xrightarrow{1,j} N$	β	
12	α	$SMOFF^0 \Rightarrow \xrightarrow{1} SINH$	$TMOFF^0 \Rightarrow$ PULSE TINH & TW DE LINE	$UMOFF^0 \Rightarrow$ PULSE UINH & UW DE LINE	$M \xrightarrow{0,1} E$		α	12
12	β	$\xrightarrow{0} SR_u$					β	
13	α	$\overline{PKM^V} \cdot PKM^{legal} \cdot NP_{38}^{er} \Rightarrow \xrightarrow{1} NPAL$	$\overline{PKM^V} \cdot PKM^{legal} \cdot NP_{38}^{er} \Rightarrow \xrightarrow{1} NPAL$	$\overline{PKM^V} \cdot PKM^{legal} \cdot NP_{38}^{er} \Rightarrow \xrightarrow{1} NPAL$			α	13
13	β						β	
14	α						α	14
15	α				$\xrightarrow{22} PK$	$\xrightarrow{22} PK$	α	15
16	α	$\xrightarrow{22} PK$	$\xrightarrow{22} PK$	$\xrightarrow{22} PK$			α	16
22	α	$\xrightarrow{0} SINH$					α	22



INTERLOCK CONTROL LEVELS

$CSI^{start} = START_2^1 \cdot CSKS^0 \cdot PI_1^0 \cdot PI_3^1 \cdot XB^0 \cdot XW^0 \cdot EB^0 \cdot PK^{00\alpha}$

$PI^{start_1} = START_2^1 \cdot PKS_1^0 \cdot PI_1^0 \cdot PI_3^0 \cdot XB^0 \cdot \overline{AL} \cdot CSK_4^0 \cdot \{QB^0 + NO^0 \cdot [\overline{(P^S \cdot Q^S) + (P^T \cdot Q^T) + (P^U \cdot Q^U) + (P^V \cdot Q^V)}]\}$

$PI^{start_2} = START_2^1 \cdot PKS_2^0 \cdot XB^0 \cdot QB^0$

$QI^{start} = START_2^1 \cdot QKS^0 \cdot PI_1^1 \cdot FI'$

$PI^{leave\,seq.} = PI^{AE\,ch\,seq} \cdot [PKIR^{AE} + (PKIR^{QK} \cdot XA^{AE})] + SS^{att\,req} \cdot PKIR^{tsd} \cdot [IOCM^{BB} + (QKIR^{tsd} \cdot QB')]$

$PI^{wait} = AEI \cdot [PKIR^{AE} + (PKIR^{QK} \cdot XA^{AE})] + PKIR^{tsd} \cdot [IOCM^{BB} + (QKIR^{tsd} \cdot QB')]$

$PI^{AE\,ch\,seq} = AEI \cdot SS^{att\,req} \cdot PI_4^0$

$PI^{ch\,seq} = PKIR_h^0 \cdot [SS^{ch\,req} + (SS^{att\,req} \cdot PKIR^{dis\,req})$

$Start\ FK = [FI_1^0 \cdot EB^0 \cdot (PKIR^{sf} + PKIR^{ja})] + (QK^{00\alpha} \cdot QI^{start} \cdot \overline{PKIR^f} \cdot \overline{PKIR^{skm}}) + [QK^{13\alpha} \cdot (QKIR^{spf} + QKIR^{spg})$

$SS^{att\,req}$ = AT LEAST ONE FLAG IS UP OTHER THAN $\overline{KD'}$

$SS^{ch\,req}$ = A FLAG OF HIGHER PRIORITY THAN $\overline{KD'}$ IS UP

$\overline{MPA} = MP_{38}^{odd} + QKM^V + QB^0 + MPS^1$

$PKM^{legal} = (SMOFF^0 \cdot PKM^S) + (TMOFF^0 \cdot PKM^T) + (UMOFF^0 \cdot PKM^U) + PKM^V$

$AEI = AEP' + (\overline{QK^{00\alpha}} \cdot QK_{5\alpha}^0 \cdot QKIR^{AESK})$

$AEB = AK_{\alpha.0}^0$

$PKIR_{op}$ CLASS DECODER LEVELS

$PKIR^{opr\,AE} \Rightarrow PKIR^{opr} \cdot N_{2.8}^0 \cdot N_{2.7}^1$

$PKIR^{ios} \Rightarrow PKIR^{opr} \cdot N_{2.8}^0 \cdot N_{2.7}^0$

$PKIR^{XM} \Rightarrow PKIR^{skm} \cdot PKIR^{jx} + (PKIR^{jmp} \cdot PKIR_{cf_2^1})$

$\overline{PKIR^{QR}} \Rightarrow PKIR_{op}^{0X} + (PKIR_{op}^{4X} \cdot PKIR_{op_3^1}) + PKIR^{skx} + \overline{PKIR^{def}}$

$PKIR^{ind} \Rightarrow PKIR_{op_6^1} + PKIR_{op_5^1} + (PKIR^{jmp} \cdot PKIR_{cf_1^1}) + [PKIR^{skx} \cdot (PKIR_{cf_2^1} + PKIR_{cf_3^1})]$

$\overline{PKIR^{def}} \Rightarrow [(PKIR_{op}^{5X} + PKIR_{op}^{0X}) \cdot PKIR_{op_3^0}] + (\overline{PKIR^{sed}} \cdot PKIR_{op}^{X3}) + PKIR^{udfz} + (N_{2.9}^0 \cdot N_{2.8}^1 \cdot PKIR^{opr})$

$PKIR^{dis\,req} \Rightarrow (PKIR_{cf_5^1} \cdot PKIR^{jmp}) + PKIR^{tsd} + (PKIR^{jx} \cdot XJ) + PKIR_{cf_5^1} \cdot \{[PKIR^{ios} \cdot (\overline{K^{eqJ}} + \overline{N_{2.6-2.4}^{101}})] + [PKIR^{skx} \cdot (\overline{K^{eqJ}} + PKIR_{cf_4^0})]\}$

$PKIR^{AE} \Rightarrow [PKIR^{def} \cdot PKIR_{op_5^1} \cdot (PKIR_{op_6^1} + PKIR_{op_3^1})] + [PKIR_{op}^{4X} \cdot (PKIR_{op}^{X1} + PKIR_{op}^{X2})] + [PKIR_{op}^{5X} \cdot (PKIR_{op}^{X5} + PKIR_{op}^{X4})] + PKIR^{ja} + PKIR^{opr\,AE}$

$PKIR^{dis} \Rightarrow PKIR^{opr} + PKIR^{jmp} + PKIR^{jx} + PKIR^{skx} + PKIR^{skm} + PKIR^{sed} + PKIR^{ja} + PKIR^{tsd}$

$PKIR^{ja} \Rightarrow PKIR^{jov} + PKIR^{jpa} + PKIR^{jna}$

$PKIR^{jx} \Rightarrow PKIR^{jpx} + PKIR^{jnx}$

$PKIR^{lf} \Rightarrow PKIR_{op}^{2X} \cdot (PKIR_{op}^{X2} + PKIR_{op}^{X1})$

$PKIR^{sf} \Rightarrow PKIR_{op}^{3X} \cdot (PKIR_{op}^{X2} + PKIR_{op}^{X1})$

$PKIR^{ff} \Rightarrow PKIR^{f} \cdot PKIR_{op}^{X2}$

$PKIR^{f} \Rightarrow PKIR^{lf} \cdot PKIR^{sf}$

PK TIMING CHART

16-4.2 PK CYCLES

16-4.2.1 INTRODUCTION

Unlike QK, there are several basic PK cycles. When PK runs, it can be obtaining an instruction or deferred-address word from memory or it can be computing the final deferred-address. Except in the latter case, some sort of memory cycle is always performed during the running of the PK cycle. During all of these basic PK cycles, certain pulses, such as the IOI clock pulses, X read pulses, etc., are always fired off.

When a new instruction is to be performed, PK first reads an instruction word out of memory. If this instruction does not call for a deferred-address, then PK immediately goes on to do whatever may be called for by the instruction. On the other hand, if a deferred address is called for, PK goes through another cycle, during which it reads out the deferred-address word from memory. If this deferred-address word calls for still another deferred-address word, the cycle is repeated. Finally a deferred-address word is obtained which does not call for another deferred-address word. PK now performs the so called ultimate deferred cycle, during which the final base address is computed. No memory cycle is involved in this step. The final base address is usually modified by two index registers rather than one.

The organization of the PK Memory timing chart reflects basic PK cycles. This is shown by the small chart abstract on the main chart. Three columns are usually examined in order to determine which events are initiated by a given PK time level. The miscellaneous control column is always examined. A decision must then be made whether or not deferred addressing is involved, and if deferred addressing is involved, which of the deferred addressing cycles is involved. After the basic PK cycle has been selected, the appropriate memory column is selected, i.e., S, T, U, V_{FF}, or $V_{\overline{FF}}$. The events occurring in any PK time level are then the sum of the events occurring in the selected columns.

16-4.2.2 PK TIME CHART

Initially PK waits in state PK^{00} until a $PI^{START}1$ level occurs. It then goes through PK^{22} and in so doing executes the basic PKM memory cycle. If the instruction word, obtained by the memory cycle, calls for a deferred-address, PK goes to PK^{00} and waits for a $PI^{START}2$ level to occur. (This wait condition also applies to the start of the ultimate deferred cycle.)

If the instruction does not call for a deferred-address, PK will proceed from PK^{22} to PK^{24} and then either go back to PK^{00} or go on to execute a PKEI cycle depending on the instruction. If a PKEI cycle is executed, PK will proceed through $PK^{31\alpha}$ and then go back to $PK^{00\alpha}$. The various PKEI cycles are discussed under the corresponding instruction headings later.

The specific events taking place while the PK counter is running will now be examined and explained. (The DSK logic will not be discussed since this is covered in detail in the discussion of the CSK timing chart.)

16-4.2.3 MISCELLANEOUS CONTROLS

The events discussed here take place in all the basic PK cycles.

The memory on-off switches are sampled in PK^{00} if QK is also in the QK^{00} state.

The IOI clock pulses to the IO units are generated in $PK^{01\alpha}$ and $PK^{12\alpha}$. These pulses are inhibited if the QK cycle of a TSD overlaps the current PK cycle. (See TSD discussion.)

The remainder of the miscellaneous control pulses involve the X Memory system. $PK^{12\alpha}$ sets the X busy (XB) interlock and initiates the X read cycle. Normally $PK^{13\alpha}$ strobes the contents of XM into X. Under certain conditions, X will be cleared instead of the strobe occurring. For example, this happens if the 00 X Memory register is selected (since this register is thought of as containing ZEROS). X is also cleared by $PK^{13\alpha}$ under the following circumstances. If a change of sequence occurs, XPS is set and a program counter is read out of X_k. After the read out occurs, X_k contains nothing of meaning. If an instruction now occurs in which the N_j bits select the X_k register ($K^{eq\ J}$), $PK^{13\alpha}$ will clear X. The $K^{eq\ J}$ condition will also cause XPS to be cleared in $PK^{15\alpha}$. Note that the XWK cycle initiated in $PK^{14\alpha}$ or later during the instruction will write something into the X_k register. Because of this the next time a $K^{eq\ J}$ type instruction occurs, XPS^0 will be true, X will not be cleared in $PK^{13\alpha}$ and XM will be strobed into X.

$PK^{15\alpha}$ also generates the X parity alarm, if an X parity condition exists (XP_{19}^{ev}).

16-4.2.4 INSTRUCTION WORD CYCLE (NO DEFERRED ADDRESSING)

As soon as the $PI^{START}1$ level is generated, PKA is set and DFA is cleared. ($PKA^1 \cdot DFA^0$ indicate to the address decoding system that PK is executing an instruction word in which the content of P selects a memory register.)

$PK^{09\alpha}$ causes a PSAL alarm, if an illegal memory is addressed by P.

The hold and OP instruction word bits in N are jammed into PKIR at $PK^{12\alpha}$. Note that $PK^{11\beta}$ is the latest time at which the memory register in which the instruction is held is strobed into N.

FI is cleared by $PK^{13\alpha}$ for certain instructions whose execution logic makes use of the F Memory during the PK cycle. (See discussion of JOV, JNA, JPA, FLF and FLG.) $FLAG_{42}$ is raised in $PK^{13\alpha}$ as part of the meta bit sensing logic of the Trapping Sequence (see Chapter 15).

For the majority of instructions, the X write counter (XWK) is started in $PK^{14\alpha}$. However for the $PKIR^{XM}$ instructions, XWK is started in $PK^{31\alpha}$ or during a DSK cycle in $PK^{23\alpha}$. The $PKIR^{IND}$ level will be jammed into XAS at $PK^{14\alpha}$ in order to determine whether or not the base address of the instruction is to be indexed.

The execution logic of JPX (06) requires that X be complemented at $PK^{15\alpha}$. (See JPX discussion.) $PK^{15\alpha}$ also causes an OCSAL alarm if PK is trying to execute an instruction with an undefined operation code.

In $PK^{22\alpha}$ a number of decisions are made to determine what activity will follow the current PK cycle. If the conditions for waiting are not present ($\overline{PI^{WAIT}}$) PK jumps from $PK^{22\alpha}$ to $PK^{24\alpha}$. If, in addition, an operand is called for by the current instruction, PI_1 is set.

If the conditions for waiting are present (PI^{WAIT}) in $PK^{22\alpha}$, either the conditions for leaving the current sequence are also present ($PI^{LEAVE\ SEQ}$), in which case PK reverts to PK^{00} and a CSK cycle occurs, or the conditions for leaving the sequence are not present ($\overline{PI^{LEAVE\ SEQ}}$), in which case a DSK cycle is initiated and PK goes to the $PK^{23\alpha}$ waiting state. (The FLAG dismissing conditions in $PK^{22\alpha}$ are discussed in conjunction with the TSD timing chart.)

$PK^{23\alpha}$ clears PKA (indicating that the memory cycle is over). PK waits in PK^{23} examining the DSK logic for a decision as to how to proceed.

$PK^{24\alpha}$ jams the hold bit into PI_4. If the current instruction does not go through $PK^{31\alpha}$ ($\overline{PKIR^{DIS}}$), then PK reverts to $PK^{00\alpha}$. If the current instruction is not an IOS, the conditions for a change of sequence ($PI^{CH\ SEQ}$) are examined at the end of the instruction. If these change sequence conditions are present, PI_3 is set as part of the CSI^{START} logic. The $PKIR^{DIS}$ type instructions examine the change sequence conditions again in $PK^{31\alpha}$ at which time IOS is included.

INSTRUCTION WORD CYCLE (DEFERRED ADDRESS). This PK cycle is identical to the previous case, except that the instruction word read out of memory calls for a deferred-address word. Hence it is always followed by an intermediate deferred-address cycle. Once the basic memory cycle (PKM) is complete (PK^{00} - PK^{22}), PK goes back to PK^{00}. The memory cycle is essentially the same as the memory cycle for the no-deferred-addressing instruction word cycle. The differences show up in the setting and use of PI_2 and PI_5.

The latest time at which the instruction is strobed into N is $PK^{11\beta}$. The defer bit ($N_{2.9}$) is examined in $PK^{13\alpha}$. If a deferred-address is called for ($N^1_{2.9}$), PI_2 is set to ONE. Assuming the instruction is defined ($PKIR^{DEF}$), PI_5 is in turn set to ONE in $PK^{14\alpha}$. The conditions ($PI^1_2 \cdot PI^1_5$) that require an intermediate deferred-address cycle to follow the current PK cycle have now been set up.

Jamming PI_5 into XAS in $PK^{14\alpha}$ prevents the base address from being indexed. For this reason the X read-write cycle, while it does occur, doesn't accomplish anything.

At the end of this cycle, as in the previous case, the configuration, hold, and OP code bits of the instruction word are contained in the $PKIR_{CF}$ and $PKIR_{OP}$ registers. The PKIR registers store these bits all through the succeeding intermediate and ultimate deferred cycles (after which they are decoded and used in the normal manner). The output of XA now specifies the address of the first intermediate deferred-address.

INTERMEDIATE DEFERRED-ADDRESS CYCLE. PK waits in PK^{00} until the $PI^{START}{}_2$ conditions are satisfied. When this occurs, the output of the X Adder is jammed into Q. DFA and PKA are also set in $PK^{00\alpha}$ when the $PI^{START}{}_2$ conditions are satisfied. $PKA^1 \cdot DFA^1$ now indicates that Q contains the address of an intermediate address and that Q can select a memory register.

If the deferred-address strobed into N has a defer bit in the ONE state ($N^1_{2.9}$), the current intermediate deferred-address cycle will be followed by another intermediate deferred-address cycle. During the first intermediate deferred cycle XAS will be in the ZERO state from PK^{00} through $PK^{14\alpha}$, $QKIR_{CF}$ will be cleared at $PK^{01\alpha}$ and the N_J bits will be jammed into $QKIR_{CF_{9-4}}$ at $PK^{06\alpha}$. XAS is set to ONE by jamming PI_5 into XAS in $PK^{14\alpha}$. In all the succeeding intermediate deferred-address cycle the content of $QKIR_{CF}$ will not be altered. The fact that XAS is now set to ONE causes the deferred base address, represented by the base address bits ($N_{2,1}$), to be indexed in all the intermediate deferred-address cycles.

Finally a deferred-address is strobed into N in which the defer bit is a ZERO ($N^0_{2.9}$). This causes PI_5 to be cleared to ZERO at $PK^{14\alpha}$, and the next PK cycle to be an ultimate deferred-address cycle.

Note that in each intermediate address cycle $FLAG_{42}$ can be raised on the $PK^{13\alpha}$ as part of the sense metabit logic of the Trapping Sequence.

ULTIMATE DEFERRED-ADDRESS CYCLE. PKA is cleared in the $PK^{00\alpha}$ state of the ultimate deferred cycle indicating a memory element register will not be strobed into N during this cycle.

PK waits in PK^{00} for the $PI^{START}2$ condition to occur. This condition allows the console stop-start control to control the start of this cycle. When $PI^{START}2$ occurs DFA is set to ONE and the output of XA is jammed into Q, but these events do not influence the operation of the computer.

The ultimate deferred-address is now formed in $N_{2,1}$ by the following steps. The contents of E is temporarily stored in M and E is cleared. The content of $QKIR_{CF_{9-4}}$ (the original J bits) is then placed in $E_{3.6\ -\ 3.1}$ and at the same time the output of the XA is placed in $E_{2,1}$.

$N_{4,2,1}$ is cleared and the content of E is loaded into N. N_J now contains the original J bits, $N_{2,1}$ contains the deferred-address formed in XA during the previous intermediate deferred cycle, and the remainder of N contains ZEROS. The original H, CF, and OP bits, however, are still in $PKIR_{CF}$ and $PKIR_{OP}$. Note that N and $PKIR_{CF}$ and $PKIR_{OP}$ are all set up by $PK^{13\alpha}$. The balance of the PK cycle is similar to the corresponding cycle for an instruction word in which there is no deferred addressing. $PK^{13\alpha}$ clears PI_2 thereby removing the last indication of the deferred addressing cycles.

16-4.2.5 MEMORY CYCLES

S MEMORY CYCLE (PKM^S). Two tapped delay lines are used to set the two read flip-flops SR_U and SR_V. The read current in the memory occurs when both flip-flops have been set. The SR_V SET delay line is pulsed at $PK^{02\beta}$ and the SR_U SET delay line is pulsed at $PK^{03\beta}$.

The write current is turned on by pulsing the SR_V and SR_U CLEAR delay lines. The SR_V CLEAR delay line is pulsed at $PK^{11\alpha}$ and the SR_U CLEAR delay line is pulsed at $PK^{12\beta}$. The write current is not actually turned on until both the SR_V and SR_U flip-flops have been cleared. The inhibit currents are turned on by pulsing the SINH SET delay line. This occurs 0.2 microsecond before SR_U is cleared. The write cycle extends from $PK^{13\alpha}$ through $PK^{22\alpha}$.

The memory word is strobed into N by pulses from the strobe delay lines. This line is pulsed at $PK^{10\beta}$. If the memory word is read out incorrectly (NP_{38}^{ev}), the parity circuits will cause an NPAL alarm in $PK^{13\alpha}$.

The reasons for clearing $N_{4,2,1}$ in $PK^{10\alpha}$ involve the execution logic for specific instructions rather than the requirements of the basic memory cycle and are discussed elsewhere in the chapter.

T AND U MEMORY CYCLES (PKM^{T} AND PKM^{U}). The T and U memory cycles are identical and are in fact very similar to the S Memory cycle. However, here the read and write currents are determined by the read and write flip-flops directly.

The T (U) read delay line is pulsed at $PK^{01\alpha}$. Pulses from this line both set and clear the TR (UR) read flip-flops. PK then jumps to $PK^{09\alpha}$. The memory register is strobed into N at $PK^{10\beta}$. The delay line is tapped so as to set the TINH (UINH) flip-flop before the TW (or UW) flip-flop. At the end of the write time, TW (UW) and then TINH (UINH) are cleared. The other PK Memory logic is identical to that found above under the S Memory cycle description.

V_{FF} MEMORY CYCLE (PKM_{VFF}). This memory cycle has several peculiarities. In the case of the S, T and U memory cycles, the word selected in memory was strobed directly into N. In the present case, N is always loaded from E. This is done by temporarily storing the content of E in M, and at the same time clearing E. The selected V_{FF} register is then loaded into E. The content of E is then loaded into N and then E is restored by transferring the content of M into E.

Certain conditions can cause PK to wait in PK^{02}. If E is busy (EB^{1}), if M is busy (QB^{1}) or if the V_{FF} Memory register is in the Arithmetic Element and the Arithmetic Element is currently busy (QB^{1} + AEB), PK must wait in $PK^{02\alpha}$. (The DSK logic which can occur in $PK^{02\alpha}$ will be discussed in conjunction with the CSK timing chart.)

Note that the content of N does not need to be rewritten in the selected V_{FF} register.

$V_{\overline{FF}}$ MEMORY CYCLE ($PKM^{V}\overline{FF}$). In this case, the selected $V_{\overline{FF}}$ register is loaded directly into N without going through E and there is no rewrite cycle. For these reasons, the $PKM^{V}\overline{FF}$ logic is very simple and consists only of the strobe pulse at $PK^{11\beta}$.

QK

16-4.3 QK CYCLES

16-4.3.1 INTRODUCTION

The QK counter runs only during those instructions that have an operand cycle. The counter's basic function is to control the operand word's memory read-write cycle. The QK cycle is always preceded by the associated PK cycle. Once the QK counter starts it always completes the entire operand cycle before another QK cycle can begin. If the operand is stored in the V_{FF} Memory it is possible that QK may have to wait in $QK^{03\alpha}$ until the interlock conditions for proceeding have been satisfied.

The QK timing chart requires that the contents of three columns be examined to determine what events are initiated in any given QK time level. The Interlock columns, the Alarm and Miscellaneous Controls columns and one of the five Memory Cycle columns must be selected and examined.

16-4.3.2 BASIC QK CYCLE

QK waits in QK^{00} until the QI^{START} level occurs. At this time QK begins counting and the following events take place:

1) The content of $PKIR_{OP}$ is copied into $QKIR_{OP}$.

2) If PK is in PK^{00}, the memory on-off switches are sampled.

3) The address of the operand used by the current instruction is copied from XA into Q.

4) Several interlock conditions are set up which indicate such things as QK has started and is running (QB^1), Q can select a memory register (QKA^1) and E is busy (EB^1).

The time level in the QK cycle at which PI_1 is cleared (PI_1^0 indicates another PK or CSK cycle can begin) depends on the specific OP code being executed. Similarly the XWK and FK counters are started at the time in the QK cycle required by the execution logic of the particular OP code.

QB is cleared in QK^{31}, anticipating by 0.4 microsecond the completion of the QK Memory cycle and the completion of the use of the Q and M registers.

When an illegal memory address is decoded (e.g., an address in a memory which is turned off), a QSAL alarm will occur at $QK^{09\alpha}$.

If the meta bit is a ONE and if the toggle switch indicating that the operator is trapping on operand word metabits is set (TM^1), then the synchronizer in the Trapping Sequence will be set to a ONE ($\xrightarrow{1} SYN_{TRAP}$) at $QK^{13\alpha}$.

Parity logic prevents M from being altered unless the $\overline{MPA}$ level is present. One of the conditions that causes the $\overline{MPA}$ to be generated is MPS^1. Since it is desirable to alter M during the read portion of the QK Memory cycle, independent of the parity logic, MPS is always set to ONE at $QK^{01\alpha}$ and cleared at $QK^{11\alpha}$ (if MPAL is not suppressed). After the memory is strobed into M (normally $QK^{11\beta}$), the MPA level will depend on factors other than MPS^1, e.g., parity conditions.

Parity is not checked in the V memories. The time at which parity is checked in the S, T and U memories depends on the specific OP code. Those OP codes which skip over $QK^{12\alpha}$ will set up the parity circuits during $QK^{13\alpha}$ and check for a parity alarm in $QK^{14\alpha}$. Most other OP codes set up the parity circuits in $QK^{12\alpha}$ and check for a parity alarm in $QK^{13\alpha}$, but some do not check the parity until $QK^{18\alpha}$. Note that the $QKIR^{LOAD}$ instructions rewrite while checking parity, while the $QKIR^{STORE}$ instructions must compute parity after checking parity before rewriting. INS is a special case and will be discussed in the INS OP Code Timing Chart.

16-4.3.3 MEMORY CYCLES

Tapped delay lines are used to set the two read flip-flops SR_U and SR_V. The read current in the memory occurs when both flip-flops have been turned on. The SR_V delay line is pulsed at $QK^{02\beta}$ and the SR_U delay line at $QK^{03\beta}$.

M is cleared in $QK^{09\alpha}$ in anticipation of the S Memory strobe into $M(SM \xrightarrow{1} M)$ at $QK^{10\beta}$.

The write current is turned on by pulsing the SR_V and SR_U CLEAR delay lines. SR_V is cleared at $QK^{11\alpha}$; SR_U is cleared at $QK^{13\beta}$ for $QKIR^{LOAD}$ instructions and at $QK^{21\beta}$ for $QKIR^{STORE}$ instructions. The write current is not actually turned on until both the SR_V and SR_U flip-flops have been cleared.

The inhibit currents are turned on by pulsing the SINH SET delay line. This occurs 0.2 microsecond before SR_U is cleared.

The write cycle ($QK^{13 \text{ or } 21}$ through QK^{31}) begins earlier for $QKIR^{LOAD}$ instructions than for $QKIR^{STORE}$ instructions, so QK jumps from QK^{23} to QK^{31} for the $QKIR^{LOAD}$ instructions and from QK^{25} to QK^{31} for the $QKIR^{STORE}$ instructions.

The same operand that is written back into memory during the WRITE portion of the memory cycle is usually copied into E by an "ultimate pulse" ($M \xrightarrow{Q1} E$). This occurs at $QK^{21\alpha}$ for all the $QKIR^{STORE}$ instructions that do not select E ($\overline{QKIR^{E}}$), and at $QK^{23\alpha}$ for all the $QKIR^{LOAD}$ instructions (except SPG) that do not select E. Note that EB is cleared ($\xrightarrow{0}$ EB) at the same time the ultimate pulse is fired off.

QK can jump states in QK^{13} through QK^{21} depending on the requirements of the OP code. These jumps are independent of memory considerations.

T AND U MEMORY CYCLES (QKM^{T} AND QKM^{U}). The T and U Memory cycles are identical and are in fact very similar to the S Memory cycle.

However, here the read and write currents are determined by the read and write flip-flops directly.

The T (U) read delay line is pulsed at $QK^{01\alpha}$. QK then jumps to $QK^{09\alpha}$. The memory is strobed into M at $QK^{11\beta}$. During this read time the TR (UR) read flip-flop is turned on and then off.

In the case of the $QKIR^{LOAD}$ instructions, the "inhibit and write" delay line is pulsed at $QK^{13\alpha}$. In the case of the $QKIR^{STORE}$ instructions the delay line is pulsed at $QK^{23\alpha}$. The delay line is tapped so as to set the TINH (UINH) flip-flop before the TW (UW) flip-flop. At the end of the write time, TW (UW) and then TINH (UINH) are cleared. The other QK logic is identical to that found above under the S Memory cycle description.

V_{FF} MEMORY CYCLE ($QKM^{V}FF$). This memory cycle has several peculiarities. In the case of the S, T and U Memory cycles, the word selected in memory was strobed into M essentially at QK^{11}. Note that up to this time E is undisturbed. In the present case it is also desirable to have the word selected in the V_{FF} Memory in M by QK^{11}. However, the route from V_{FF} to M is through E, and the original content of E must be momentarily displaced and then replaced in E by QK^{11}. This is accomplished as follows:

At $QK^{02\alpha}$ the content of E is copied into M, and E is cleared.

At this point the execution logic depends on which of two cases exist. In Case 1, the selected V_{FF} register is not E ($\overline{VMD^E}$); in Case 2, the selected V_{FF} register is E.

Consider Case 1. When the waiting state logic in $QK^{03\alpha}$ permits, the content of the selected V_{FF} Memory (A,B,C or D) is copied into E. QK now jumps to $QK^{09\alpha}$ where M and E are interchanged. M now contains the content of the selected V_{FF} register and E contains its own original content.

In Case 2, E is cleared and M contains the original content of E (i.e., the content of the selected V_{FF} register) at the end of $QK^{02\alpha}$. Nothing happens in $QK^{03\alpha}$ and QK jumps from $QK^{03\alpha}$ to $QK^{09\alpha}$. M is now copied into E. Both M and E now contain the same thing, i.e., the content of the selected V_{FF} register.

M is not cleared in $QK^{09\alpha}$. As a result any operation on the metabit uses the metabit left there by the previous QK Memory cycle.

Note that no read or write cycles in the sense of the S, T and U Memory are involved in the V memories. Everything is accomplished by simple register transfers. For this reason no parity checking or parity computing is involved. QK therefore jumps from QK^{11} to QK^{13}.

The write cycle, which occurs during $QKIR^{STORE}$ type instructions, is complicated by the question of whether or not a STE instruction is being performed. As before, there are two cases. In Case 1, the selected register is not E ($\overline{VMD^E}$); in Case 2, the selected V_{FF} register is E.

In Case 1, the content of M is copied into E from M and, if this is a STE, E is saved in M at $QK^{21\alpha}$. The selected V_{FF} register (A,B,C or D) is cleared at $QK^{22\alpha}$ and E copied into the register at QK^{23}. In all instructions except SPG the ultimate pulse copies M into E at QK^{23}. At this time, if this is a STE, E is also reset from M.

In Case 2, nothing occurs at QK^{21} unless a STE is being performed. The content of M and E are interchanged if this is a STE. Nothing happens in $QK^{22\alpha}$, but in QK^{23} the content of M is copied back into E again only if this is a STE instruction which selects E.

$V_{\overline{FF}}$ MEMORY CYCLE ($QKM^{V\overline{FF}}$). In this memory cycle, M is cleared in $QK^{09\alpha}$ as usual and the V toggle switch memory register is copied into M in $QK^{11\alpha}$.

Since no write cycle is involved and there are no parity checking requirements, QK jumps from QK^{11} to QK^{13} and from QK^{23} to QK^{31}.

The ultimate pulse logic is the same as that for the S, T and U memories.

QK TIME LEVELS		INDEPENDENT OF QK MEMORY: INTERLOCKS	INDEPENDENT OF QK MEMORY: ALARMS & MISCELLANEOUS CONTROLS	QK MEMORY CYCLE: S MEMORY CYCLE (QKM^S)	QK MEMORY CYCLE: T MEMORY CYCLE (QKM^T)	QK MEMORY CYCLE: U MEMORY CYCLE (QKM^U)	QK MEMORY CYCLE: V^{FF} MEMORY CYCLE (QKM^{VFF})	QK MEMORY CYCLE: $V^{\overline{FF}}$ MEMORY CYCLE ($QKM^{V\overline{FF}}$)	QK TIME LEVELS	
00	α	$QI^{start}\cdot\overline{PKIR^{lf}}\cdot\overline{PKIR^{IX}_{op}} \supset \xrightarrow{0} PI_1$ $QI^{start} \supset \xrightarrow{1} QB$ " $\supset \xrightarrow{1} EB$ " $\supset \xrightarrow{1} QKA$ $\overline{QI^{start}} \supset \xrightarrow{0} QKA$ $QI^{start}\cdot\overline{PKIR^{f}}\cdot\overline{PKIR^{skm}} \supset \xrightarrow{start} FK$	$\overline{QI^{start}} \supset \overline{QK'}+1 \not\rightarrow QK$ $QI^{start} \supset PKIR_{op} \xrightarrow{j} QKIR_{op}$ $PK^{00\alpha} \supset SM^{off} \xrightarrow{j} SMOFF$ " $\supset TM^{off} \xrightarrow{j} TMOFF$ " $\supset UM^{off} \xrightarrow{j} UMOFF$	$QI^{start} \supset XA \xrightarrow{j} Q$	$QI^{start} \supset XA \xrightarrow{j} Q$	$QI^{start} \supset XA \xrightarrow{j} Q$	$QI^{start} \supset XA \xrightarrow{j} Q$	$QI^{start} \supset XA \xrightarrow{j} Q$	α	00
01	α	$\xrightarrow{1} MPS$			$TMOFF^{0} \supset$ PULSE TR DE LINE $\xrightarrow{09} QK$	$UMOFF^{0} \supset$ PULSE UR DE LINE $\xrightarrow{09} QK$			α	01
02	α						$\overline{MPA} \supset \xrightarrow{0} E$ $\overline{MPA} \supset E \xrightarrow{0,1} M$	$\xrightarrow{09} QK$	α	02
	β			$SMOFF^{0} \supset \xrightarrow{1} SR_v$					β	
03	α						$\overline{VMD^E}\cdot\overline{AEB} \supset VFFM \xrightarrow{1} E$ $VMD^{AE}+\overline{AEB} \supset \xrightarrow{09} QK$ $\overline{QK'}+1 \not\rightarrow QK$		α	03
	β			$SMOFF^{0} \supset \xrightarrow{1} SR_u$					β	
04	α								α	04
05	α								α	05
06	α			$\xrightarrow{09} QK$					α	06
09	α	$QKIR^{IX}_{op}\cdot QKIR^{X6}_{op} \supset \xrightarrow{0} PI_1$	$\overline{QKM^{legal}} \supset \xrightarrow{1} QSAL$	$\overline{MPA} \supset \xrightarrow{0} M$	$\overline{MPA} \supset \xrightarrow{0} M$	$\overline{MPA} \supset \xrightarrow{0} M$	$M \xrightarrow{0,1} E$ $\overline{VMD^E}\cdot\overline{MPA} \supset E \xrightarrow{0,1} M$	$\overline{MPA} \supset \xrightarrow{0} M$	α	09
10	β			$SM \xrightarrow{1} M$					β	10
11	α	$\overline{MPAL_{sup}} \supset \xrightarrow{0} MPS$		$\xrightarrow{0} SR_v$ $\overline{QKIR^{flg}}\cdot\overline{QKIR^{flf}}\cdot\overline{QKIR^{ins}}\cdot\overline{QKIR^{st}} \supset \xrightarrow{13} QK$	$\overline{QKIR^{flg}}\cdot\overline{QKIR^{flf}}\cdot\overline{QKIR^{ins}}\cdot\overline{QKIR^{st}} \supset \xrightarrow{13} QK$	$\overline{QKIR^{flg}}\cdot\overline{QKIR^{flf}}\cdot\overline{QKIR^{ins}}\cdot\overline{QKIR^{st}} \supset \xrightarrow{13} QK$	$\xrightarrow{13} QK$	$\xrightarrow{13} QK$	α	11
	β				$TM \xrightarrow{1} M$	$UM \xrightarrow{1} M$		$VTM \xrightarrow{1} M$	β	
12	α								α	12
13	α	$QKIR^{spf}+QKIR^{spg} \supset \xrightarrow{start} FK$	$\overline{QKM^V}\cdot QKM^{legal}\cdot(QKIR^{fl}+QKIR^{ins}+QKIR^{st})\cdot MP^{ev}_{38} \supset \xrightarrow{1} MPAL$	$SMOFF^{0}\cdot QKIR^{load} \supset \xrightarrow{1} SINH$ $QKIR^{tsd} \supset \xrightarrow{18} QK$	$TMOFF^{0}\cdot QKIR^{load} \supset$ PULSE TINH & TW DE LINE $QKIR^{tsd} \supset \xrightarrow{18} QK$	$UMOFF^{0}\cdot QKIR^{load} \supset$ PULSE UINH & UW DE LINE $QKIR^{tsd} \supset \xrightarrow{18} QK$	$QKIR^{tsd} \supset \xrightarrow{18} QK$	$QKIR^{tsd} \supset \xrightarrow{18} QK$	α	13
	β			$QKIR^{load} \supset \xrightarrow{0} SR_u$					β	
14	α	$QKIR^{IX}_{op}\cdot\overline{QKIR^{adx}} \supset \xrightarrow{0} PI_1$	$\overline{QKM^V}\cdot QKM^{legal}\cdot(QKIR^{load}+QKIR^{com})\cdot MP^{ev}_{38} \supset \xrightarrow{1} MPAL$ $M'_{4.10}\cdot TM' \supset \xrightarrow{1}$ SYN TRAP	$QKIR^{skm}+QKIR^{ins} \supset \xrightarrow{18} QK$ $QKIR^{st}+QKIR^{load} \supset \xrightarrow{21} QK$	$QKIR^{skm}+QKIR^{ins} \supset \xrightarrow{18} QK$ $QKIR^{st}+QKIR^{load} \supset \xrightarrow{21} QK$	$QKIR^{skm}+QKIR^{ins} \supset \xrightarrow{18} QK$ $QKIR^{st}+QKIR^{load} \supset \xrightarrow{21} QK$	$QKIR^{skm}+QKIR^{ins} \supset \xrightarrow{18} QK$ $QKIR^{st}+QKIR^{load} \supset \xrightarrow{21} QK$	$QKIR^{skm}+QKIR^{ins} \supset \xrightarrow{18} QK$ $QKIR^{st}+QKIR^{load} \supset \xrightarrow{21} QK$	α	14
15	α								α	15
16	α	$QKIR^{adx} \supset \xrightarrow{0} PI_1$							α	16
17	α			$QKIR^{com} \supset \xrightarrow{21} QK$	$QKIR^{com} \supset \xrightarrow{21} QK$	$QKIR^{com} \supset \xrightarrow{21} QK$	$QKIR^{com} \supset \xrightarrow{21} QK$	$QKIR^{com} \supset \xrightarrow{21} QK$	α	17
18	α		$\overline{QKM^V}\cdot QKM^{legal}\cdot\overline{QKIR^{ins}}\cdot MP^{ev}_{38} \supset \xrightarrow{1} MPAL$						α	18
19	α								α	19
20	α								α	20
21	α	$\overline{QKM^{VFF}}\cdot QKIR^{store} \supset \xrightarrow{0} EB$ $PKIR^{lf} \supset \xrightarrow{0} PI_1$		$QKIR^{store}\cdot SMOFF^{0} \supset \xrightarrow{1} SINH$ $QKIR^{store}\cdot\overline{QKIR^E} \supset M \xrightarrow{0,1} E$	$TMOFF^{0}\cdot QKIR^{store} \supset$ PULSE TINH & TW DE LINE $QKIR^{store}\cdot\overline{QKIR^E} \supset M \xrightarrow{0,1} E$	$UMOFF^{0}\cdot QKIR^{store} \supset$ PULSE UINH & UW DE LINE $QKIR^{store}\cdot\overline{QKIR^E} \supset M \xrightarrow{0,1} E$	$QKIR^{ste}\cdot\overline{MPA} \supset E \xrightarrow{0,1} M$ $(QKIR^{store}\cdot\overline{QKIR^E})+(QKIR^{ste}) \supset M \xrightarrow{0,1} E$	$QKIR^{store}\cdot\overline{QKIR^E} \supset M \xrightarrow{0,1} E$	α	21
	β			$QKIR^{store} \supset \xrightarrow{0} SR_u$					β	
22	α	$QKIR^{ld}\cdot QKIR^{X} \supset \xrightarrow{start} XWK$					$QKIR^{store}\cdot(VMD^A+VMD^B+VMD^C+VMD^D) \supset \xrightarrow{0} VFFM$		α	22
23	α	$\overline{QKIR^{spg}} \supset \xrightarrow{0} EB$		$\overline{QKIR^{spg}}\cdot QKIR^{load}\cdot\overline{QKIR^E} \supset M \xrightarrow{0,1} E$ $QKIR^{load} \supset \xrightarrow{31} QK$	$\overline{QKIR^{spg}}\cdot QKIR^{load}\cdot\overline{QKIR^E} \supset M \xrightarrow{0,1} E$ $QKIR^{load} \supset \xrightarrow{31} QK$	$\overline{QKIR^{spg}}\cdot QKIR^{load}\cdot\overline{QKIR^E} \supset M \xrightarrow{0,1} E$ $QKIR^{load} \supset \xrightarrow{31} QK$	$(\overline{QKIR^{spg}}\cdot QKIR^{load}\cdot\overline{QKIR^E})+(QKIR^{ste}\cdot\overline{VMD^E}) \supset M \xrightarrow{0,1} E$ $QKIR^{store}\cdot(VMD^A+VMD^B+VMD^C+VMD^D) \supset E \xrightarrow{1} VFFM$ $\xrightarrow{31} QK$	$\overline{QKIR^{spg}}\cdot QKIR^{load}\cdot\overline{QKIR^E} \supset M \xrightarrow{0,1} E$ $\xrightarrow{31} QK$	α	23
24	α								α	24
25	α			$\xrightarrow{31} QK$	$\xrightarrow{31} QK$	$\xrightarrow{31} QK$			α	25
31	α	$\xrightarrow{0} QB$ $QKIR^{aux} \supset \xrightarrow{start} XWK$		$\xrightarrow{0} SINH$					α	31

$QKIR_{op}$ CLASS DECODER LEVELS

$$QKIR^{A} = QKIR^{2X}_{op} + QKIR^{3X}_{op} + (QKIR_{op^1_6}\cdot QKIR_{op^1_4})\cdot QKIR^{X4}_{op}$$
$$QKIR^{B} = QKIR^{2X}_{op} + QKIR^{3X}_{op} + QKIR^{5X}_{op}\cdot QKIR^{X5}_{op}$$
$$QKIR^{C} = QKIR^{2X}_{op} + QKIR^{3X}_{op}\cdot QKIR^{X6}_{op}$$
$$QKIR^{D} = [(QKIR^{2X}_{op} + QKIR^{3X}_{op})\cdot QKIR^{X7}_{op}] + (\overline{QKIR^{A}}\cdot QKIR^{AK})$$
$$QKIR^{E} = [(QKIR^{2X}_{op} + QKIR^{3X}_{op})\cdot QKIR^{X0}_{op}] + QKIR^{ite}\cdot QKIR^{sed}$$
$$QKIR^{AK} = QKIR_{op^1_6}\cdot QKIR_{op^1_5}$$
$$QKIR^{AESK} = QKIR^{AK}\cdot\overline{QKIR^{asa}}\cdot\overline{QKIR^{X7}_{op}}$$
$$QKIR^{store} = QKIR^{3X}_{op} + QKIR^{5X}_{op} + (QKIR^{1X}_{op}\cdot QKIR_{op^1_3})$$
$$QKIR^{load} = \overline{QKIR^{store}}$$
$$QKIR^{X} = QKIR^{1X}_{op}\cdot[QKIR^{X1}_{op} + (QKIR_{op^1_3}\cdot QKIR_{op^0_1})]$$
$$QKIR^{st} = (QKIR^{1X}_{op}\cdot QKIR^{X6}_{op}) + [(QKIR^{1X}_{op} + QKIR^{5X}_{op})\cdot QKIR^{X4}_{op}] + [QKIR^{3X}_{op}\cdot(QKIR_{op^1_3} + QKIR^{X0}_{op})]$$
$$QKIR^{ld} = QKIR^{AK} + QKIR^{aux} + QKIR^{ita} + QKIR^{una} + [(QKIR^{1X}_{op} + QKIR^{5X}_{op})\cdot QKIR^{X4}_{op}] + [QKIR^{2X}_{op}\cdot(QKIR_{op^1_3} + QKIR^{X0}_{op}) + (QKIR^{1X}_{op}\cdot QKIR^{X1}_{op})$$
$$QKIR^{specify} = QKIR^{spg} + QKIR^{spf}$$
$$QKIR^{file} = QKIR^{flg} + QKIR^{flf}$$
$$\xrightarrow{1} MPAL = [(QKIR^{file} + QKIR^{ins} + QKIR^{st})\cdot QK^{13\alpha}] + [(QKIR^{com} + QKIR^{load})\cdot QK^{14\alpha}] + (QKIR^{ins}\cdot QK^{18\alpha})$$

INTERLOCK CONTROL LEVELS

$$QI^{start} = START^1_2\cdot QKS^0\cdot PI^1_1\cdot FI^1$$
$$QKM^{legal} = (SMOFF^0\cdot QKM^S) + (TMOFF^0\cdot QKM^T) + (UMOFF^0\cdot QKM^U) + QKM^V$$
$$MPA = MP^{ev}_{38}\cdot QKM^V\cdot QB^0\cdot MPS^0$$
$$AEI = AEP^1 + (\overline{QK^{00\alpha}}\cdot QKIR^{AESK}\cdot QK^0_{5\alpha}$$
$$AEB = AK^0_{\alpha.0}$$

QK TIMING CHART

16-5 PK-QK INSTRUCTION CYCLES

16-5.1 INTRODUCTION

The timing charts in this section tabulate the sequence of logic used in the execution of specific instructions. These timing charts fall into three basic classes:

Class A. Instructions which do not use an operand. The instruction timing charts cover the PKEI (PK extended instruction) cycle (PK^{25} through PK^{31}).

Class B. Instructions which use an operand and have an extended PK cycle, i.e., a PKEI cycle. The instruction timing charts cover the PKEI cycle (PK^{25} through PK^{31}) and the operand instruction cycle (QK^{00} through QK^{31}).

Class C. Instructions which use an operand and have a simple PK cycle. The instruction timing charts cover the operand instruction cycle (QK^{00} through QK^{31}).

While there are some 35 OP codes discussed in this section, there are certain sequences of pulses which appear in more than one OP code. For example, in all load type instructions there is a sequence of pulses that initiates the configuration-sign extension pattern. Isolated pulses also occur which serve a common purpose in the instruction in which they are fired off. For example, in Class B and C instructions an "ultimate pulse" is fired off in $QK^{23\alpha}$. This pulse copies the operand written into the Memory Element register also into the E register.

The basic pattern of configuration pulses is very similar for all the configurable instructions, whether of a load or store type. An inverse permutation pulse is fired off in $PK^{11\beta}$. If a LOAD type instruction is involved, the content of M is transferred into E under permuted activity control in $PK^{13\alpha}$. If a STORE type instruction is involved, the content of E is transferred into M under permuted activity control in $PK^{13\alpha}$. A direct permutation pulse is then fired off in $PK^{13\beta}$.

The brief discussion accompanying each timing chart in this section consists of three parts:

OP CODE DESCRIPTION	-	a simple non-rigorous description of the OP code is given.
SPECIAL FEATURES	-	a list of the special features or unusual characteristics of the instruction is given, e.g., special logical transfers, special sampling nets, special counter activity, etc.
DETAILS	-	a discussion of the significance of the timing chart pulses is given.

The types of figures used to illustrate the timing charts or supplement the discussion vary in format somewhat depending on the OP code. Most figures contain two types of information. A picture is given illustrating the register transfers that occur during the instruction execution. The circled numbers on these illustrations indicate the relative order in which the transfers occur, e.g., 3 occurs after 2 and before 4. The heavy arrows indicate the transfers of major significance in the logic. The lighter arrows indicate transfers that may restore registers or provide some secondary function. The tables accompanying the figures, give the content of the registers as a result of the pulses fired off by the indicated time level. Lower case letters have been used to indicate the content of the registers before the instruction execution phase begins. For example, b represents the original content of register B, m the original content of register M, y the content of the selected Memory Element register, etc. The following symbology has also been used:

- $b_{\bar{p}}$ - the register contains the inverse permutation of the original content of B.
- y_{a_p} - the original operand has been transferred into the register under permuted activity control.
- y_{CF} - the register contains the configured operand.
- $y_{CF_{SE}}$ - the register contains the configured operand with its sign extended.
- $b|y_{CF}$ - the inactive quarters of the register contain the original contents of the corresponding quarters of B; the active quarters contain the configured operand.

In some cases the specific contents of each quarter of the register have been spelled out to indicate more clearly what is taking place. For example,

y	=	y4	y3	y2	y1

etc.

The OP codes fall in several broad categories. Within a category the OP codes have certain common features or functional similarities. There also exists within the broad categories subcategories where the similarity of OP codes becomes even more striking. The broad categories are:

OPR (IOS and AE) - in these instructions the address bits have a special function.

X Memory Instructions (JPX and JNX; AUX; RSX; ADX; DPX; and EXX) - in these instructions either a logical decision to skip or jump is made, based on the contents of one of the X Memory registers, or a load or store operation (simple or complex) occurs that involves the X Memory.

F Memory Instructions (FLF, FLG, SPF and SPG) - these are load and store instructions involving the F Memory.

Load and Store Type Instructions Involving the Arithmetic Element Registers - several sub-groupings of these instructions can be made depending on what features are being examined, e.g.,

1) LDA, LDB, LDC, LDD
2) LDA, ITA, UNA, EXA
3) STA, STB, STC, STD
4) STA, INS
5) LDA, STA, EXA, LDB, STB, LDC, STC, LDD, STD, etc.

In some of these instructions, the load or store logic is both complex and obscure, e.g., INS.

Load and Store Type Instructions Involving the E Register (LDE, ITE, STE)

Jump on Arithmetic Element Type Instructions (JOV, JPA, JNA) - these are logical instructions which sample registers or flip-flops and make jump decisions based on their contents.

COM, TSD and SKM are somewhat unique instructions, although the pattern of their execution logic is found in other instructions. TSD is similar to the LD- and ST- instructions depending on whether an input or output unit is involved. COM has some of the characteristics of both the LDE and STE instructions.

Undefined Operation Codes The operation code values 00, 01, 02, 03, 04 $\cdot$ $N^{1}_{2.8}$, 13, 23, 33, 45, 50, 51, 52, 53, 63, 73 are undefined. All of these operation codes will cause OCSAL to be set at $PK^{15\alpha}$ of this instruction word cycle (the initial PK cycle if no deferred address is requested or the ultimate PK cycle if a deferred address is requested). PK will then return to $PK^{00\alpha}$ from $PK^{24\alpha}$ and wait for the alarm condition to be removed if this condition stopped the computer. Note that when $\overline{K}^{eq\ J}$ that the undefined operation codes have no effect on the computer other than to advance the content of P by one (if $K^{eq\ J}$ then XPS is cleared).

OP CODE DESCRIPTION. The function of this OP code is determined by the base address portion of the instruction word, i.e., $N_{2.8 - 1.1}$. The basic differentiation in function is determined by $N_{2.8 - 2.7}$. Thus,

$N_{2.8 - 2.7}$		INSTRUCTION
0	0	IO OPERATION (IOS)
0	1	AE OPERATION (AOP)
1	0	UNDEFINED
1	1	UNDEFINED

If either of the two undefined instructions are executed an OCSAL alarm will be generated at $PK^{15\alpha}$.

These instructions have an extended PK cycle (instruction execution phase), and no QK cycle, i.e., no operand is obtained from memory.

The IOS instruction is discussed in detail in Chapter 15 on the In-Out Element, while AOP instruction is discussed at length in Chapter 14 on the Arithmetic Element. Brief discussions of the execution logic of each instruction is included in the descriptions of the IOS and AOP timing chart.

OPERATE (IN-OUT SELECTION: $N^0_{2.8} \cdot N^0_{2.7}$)

OP CODE DESCRIPTION. IOS is used to control and/or report on the state of the In-Out system, as well as to raise and lower flags in the Sequence Selector. It is one of the variations of the OPR instruction. It is also a non-configurable and non-indexable type instruction.

SPECIAL FEATURES. IOS has an extended PK cycle (PK^{25} through PK^{31}) and no QK cycle. The base address bits $N_{2.6-1.1}$ are used to determine the type of IOS executed. The significance of the instruction word bits is shown in the accompanying table. Note that only CF_5 and CF_1 of the configuration bits are used, and that $N^{00}_{2.8-2.7}$ distinguish this as an OPR^{IOS} instruction.

OPR^{IOS} 04

DETAILS. Since an IOS 3X,XXX or 6X,XXX is used to change the operating mode of an IO unit, an IOSAL will occur if the selected unit is in the MAINTenance mode at $PK^{24\alpha}$.

In a "report" type IOS (CF^1_1), E is cleared preliminary to the transfer of information into E. This clearing occurs in $PK^{25\alpha}$ when E is not busy (EB^0).

$PK^{25\alpha}$ is a waiting state and depends on the following conditional logic:

$$PK^{25\alpha}\,(EB^1 + QB^1) \supset \overline{PK}\rceil \;+\; 1 \not\longrightarrow PK$$

E must be free, since an IOS "report" into E will occur in $PK^{26\alpha}$ if CF^1_1. It is also important that the current IOS not upset the mode of the IO unit of the current sequence, since a data transfer during the QK cycle of a TSD can still be taking place. Also, if the IOS refers to the Trapping Sequence, it must not change the set meta bit mode during a QK cycle. For these reasons IOS is held up in $PK^{25\alpha}$ until the previous QK cycle is completed (QB^0) or in the case of SPG, until the FK cycle is completed. (FK clears EB in this case, instead of QB.)

The information placed in bits 3.6 - 3.1 of E is simply the number of the specified IO unit. The information placed in the remaining bits of E report on the situation at the IO unit and in the Sequence Selector before this situation is changed by the IOS.

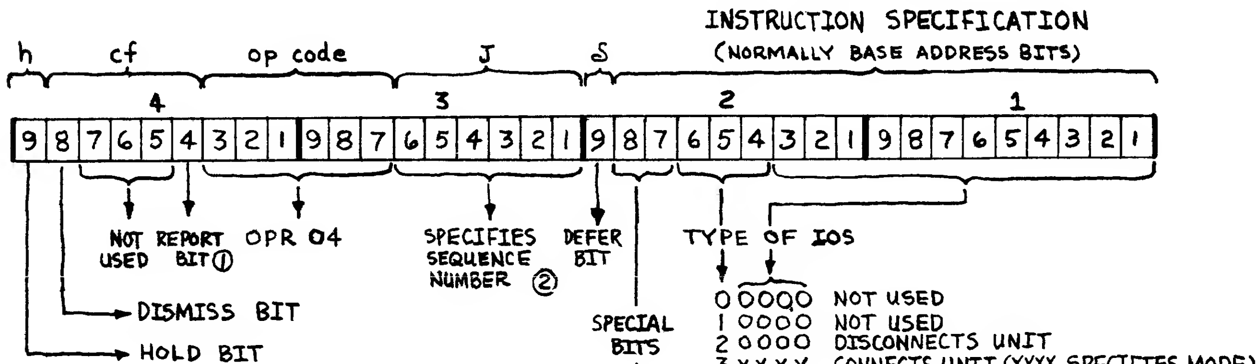
The new modes specified by the IOS type are established by pulses generated at $PK^{26\alpha}$. Note that changes in the operating mode of an IO unit are prevented if the specified IO unit is in the MAINTenance state.

A complete discussion of the effect of 20000, 3XXXX and 6XXXX is given in Chapter 15 under each IO unit.

If CF_5^1 the $PKIR^{DIS\ REQ}$ level will be generated and, if the hold bit is a zero, a dismiss will occur. The flag will be lowered in $PK^{25\alpha}$, and either PI_3 or CSK_4 will be set in PK^{31}. However, the IOS will not dismiss if it also raises the flag of the current sequence $(K^{eq\ J} \cdot N^{101}_{2.6\ -\ 2.4})$. In this case the flag lowering in PK^{25} is offset by the flag raising in PK^{26}, so that in fact the flag is left raised at the end of the instruction. PK goes through four states (1.6 microseconds) after PK^{26} before sampling the interlock nets in order to allow them to set up properly after the mode pulses.

OPERATE (IN-OUT SELECTION: $N^{0}_{2.8} \cdot N^{0}_{2.7}$)

	24	α	$IOCM^{maint} \cdot (N^{110}_{2.6-2.4} + N^{011}_{2.6-2.4}) \cdot \ldots \supset$	$\xrightarrow{1} IOSAL$
	25	α	$EB^{1} + QB^{1} \cdot \ldots \supset$	$\overline{PK}' + 1 \rightarrow PK$
			$EB^{0} \cdot PKIR_{cf_1}' \cdot \ldots \supset$	$\xrightarrow{0} E$
			$PKIR^{0}_{h} \cdot PKIR^{dis\ req} \cdot \overline{KD^{00}} \cdot \ldots \supset$	$\xrightarrow[KD]{dismiss} Flag$
PK	26	α	$PKIR_{cf_1}' \cdot \ldots \supset$	$IOBM \rightarrow E;\ N_{3.6-3.1} \xrightarrow{1} E_{3.6-3.1}$
			$N^{010}_{2.6-2.4} \cdot \ldots \supset$	$\xrightarrow[ND]{0} C$
			$(N^{011}_{2.6-2.4} + N^{110}_{2.6-2.4}) \cdot \overline{IOCM^{maint}} \cdot \ldots \supset$	$\xrightarrow[ND]{Mode + Select} IOC$
			$N^{100}_{2.6-2.4} \cdot \ldots \supset$	$\xrightarrow[ND]{0} Flag$
			$N^{101}_{2.6-2.4} \cdot \ldots \supset$	$\xrightarrow[ND]{1} Flag$
	27	α		
	28	α		
	29	α		$\xrightarrow{31} PK$
	31	α	$PI^{ch\ seq} \cdot \ldots \supset$	$\xrightarrow{1} PI_3$
			$PKIR^{0}_{h} \cdot PKIR^{dis\ req} \cdot \overline{SS^{att\ req}} \cdot \ldots \supset$	$\xrightarrow{1} CSK_4$



0 0000 NOT USED
1 0000 NOT USED
2 0000 DISCONNECTS UNIT
3 XXXX CONNECTS UNIT (XXXX SPECIFIES MODE)
4 0000 LOWERS FLAG
5 0000 RAISES FLAG
6 XXXX SELECTS SUB UNIT (XXXX SPECIFIES SUB UNIT)
7 0000 NOT USED

00 = IOS

① "REPORT" IS INDEPENDENT OF THE REST OF THE INSTRUCTION.

REPORTS ARE:

$E_{2.9}$ = FLAG
$E_{2.8}$ = BUFFER STATUS
$E_{2.7}$ = MAINTENANCE
$E_{2.6}$ = CONNECT
$E_{2.5}$ = $C' \cdot$ EIA
$E_{2.4}$ = $C' \cdot$ MISIND

② VALUES 1 THRU 37 CAUSE THE INSTRUCTION TO BE IGNORED. ONLY VALUES 0 AND 40 THRU 77 ARE ACCEPTABLE.

OP CLASS DECODER LINES UP:

$PKIR_{cf_5}' \cdot (K^{eqJ} \cdot N^{101}_{2.6-2.4}) \cdot \supset PKIR^{dis\ req},\ PKIR^{def},\ PKIR^{dis}$

$\xrightarrow[ND]{Select} IOC = N^{110}_{2.6-2.4} \cdot PKIR^{ios}$

$IOC \xrightarrow[ND]{} E = \overline{IOB \xrightarrow[KD]{} E}$

$CSK_4' \cdot PK^{odd} \cdot K_{3.6}' \supset KD \neq K$

($K_{3.6}$ IS MADE TO LOOK LIKE A ZERO INTO THE K DECODER)

NOTE: COMPUTER PRESET WILL CLEAR C. THE TRANSITION OF C TO THE "ONE" STATE WILL CLEAR MISIND & STATUS IN AN INPUT UNIT AND CLEAR MISIND AND SET STATUS IN AN OUTPUT UNIT.

C.A.N. 7-11-61

OPERATE (ARITHMETIC ELEMENT: $N^{0}_{2.8} \cdot N^{1}_{2.7}$)

OP CODE DESCRIPTION. AOP allows the programmer to operate on existing data in the Arithmetic Element with any one of a number of defined AK type instructions without obtaining an operand from memory. It is one of the variations of the OPR instruction. AOP is a non-configurable and non-indexable type instruction.

SPECIAL FEATURES. AOP has an extended PK cycle (PK^{25} through PK^{31}) and no QK cycle. The base address bits $N_{2.6-1.1}$ are used to determine the specific AK type instruction executed. The significance of the instruction word bits is shown in the accompanying table. Note that the $N_{4.8-4.4}$ bits are not used, and that $N^{01}_{2.8-2.7}$ distinguish this as an OPR^{AE} instruction.

OPR^{AE}
04

DETAILS. $PK^{25\alpha}$ is a waiting state conditioned by the following logic:

$$QB^{1} + AEB \supset \overline{PK}\rceil + 1 \longrightarrow PK$$

QB^{1} and AEB simply insure that the QK and AK cycles of any previous instructions are finished. The $N_{2.6-1.4}$ bits are also jammed into $AKIR_{CF}$ and $AKIR_{OP}$ in $PK^{25\alpha}$.

$PK^{26\alpha}$ initiates the AK counter which executes the instruction logic specified by the contents of $AKIR_{CF}$ and $AKIR_{OP}$.

If an undefined Arithmetic Element operation code is specified in $AKIR_{OP}$ by AOP, an OCSAL alarm will occur during the AK cycle.

Note that the configuration used in the Arithmetic Element during an AOP is specified directly by the $N_{1.9-1.4}$ bits. No permutation is specified, since permutation has meaning only in the Exchange Element. Also, since the configuration is not transmitted through $QKIR_{CF}$ to $AKIR_{CF}$, no activity or sign extension will occur in partially active subwords. Hence operations specified by AOP with such configurations will yield different results than when specified in the usual manner.

OPERATE (ARITHMETIC ELEMENT: $N^{0}_{2.8} \cdot N^{1}_{2.7}$)

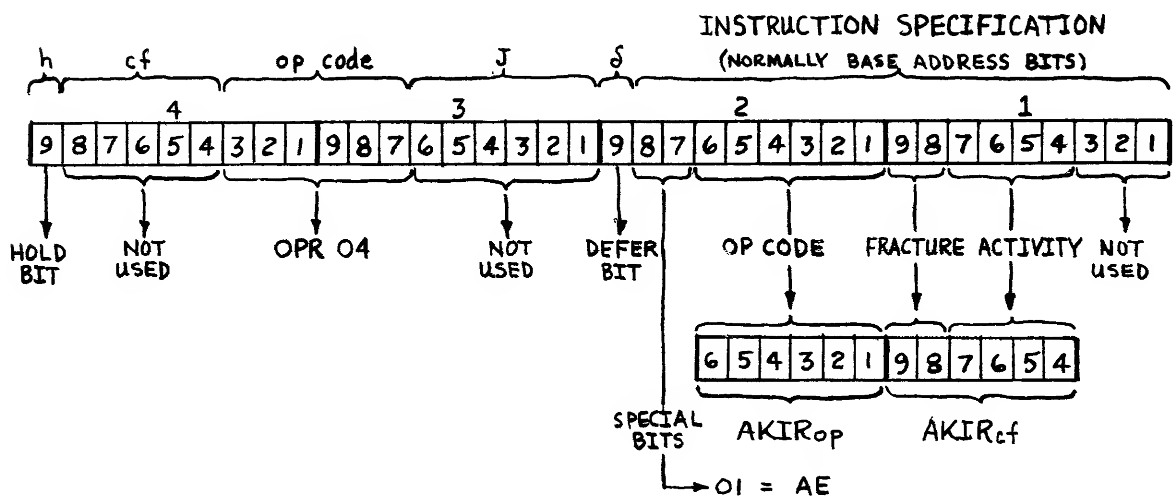
PK

24	α		
25	α	$QB' + AEB$ . . . ⊃	$\overline{PK}+1 \to PK$
		AK^{1}_{0} . . . . ⊃	$\xrightarrow{0} AK_{11-1}$, $\xrightarrow{1} AK_{0}$
		" . . . . ⊃	$N_{2.6-2.1} \xrightarrow{j} AKIR_{op_{6-1}}$
		" . . . . ⊃	$N_{1.9-1.4} \xrightarrow{j} AKIR_{cf_{9-4}}$
26	α		$\xrightarrow{31}$ PK
			$\xrightarrow{1}$ AEP
			$\xrightarrow{start}$ AK
31	α	$PI^{ch\ seq}$ . . . . ⊃	$\xrightarrow{1} PI_{3}$

OP Class Decoder Lines Up: $PKIR^{def}$
$PKIR^{dis}$
$PKIR^{AE}$

$AEB \cdot \cdot \cdot = AK^{0}_{0}$

$PKIR^{opr\ AE} = PKIR^{opr} \cdot N^{0}_{2.8} \cdot N^{1}_{2.7}$



OP CODE DESCRIPTION. JMP performs an "unconditional jump" to the memory address specified by the output of the X Adder. After P is indexed in the normal manner, the content of P is replaced by the content of the X Adder. Before this occurs, the content of P and Q may be placed in the E register. JMP is a non-indexable and non-configurable instruction.

SPECIAL FEATURES. JMP has an extended PK cycle (PK^{25} through PK^{31}) and no QK cycle. All of the configuration bits are used for special purposes: e.g., the $PKIR^{DIS\ REQ}$ (dismiss request instruction), $PKIR^{IND}$ (indexing instruction) and $PKIR^{XM}$ (X Memory instruction) class decoder levels are dependent on $PKIR_{CF}$.

DETAILS. After the content of the X Memory (x_j) is strobed into X in $PK^{13\alpha}$ (as it is in all other instructions), the following actions, conditional on the $PKIR_{CF}$ bits, take place:

CF_1^1. The output of the X Adder equals the arithmetic expression ($y + cf_1 \cdot x_j$), i.e., the content of $N_{2,1}$ is indexed by x_j only if CF_1^1.

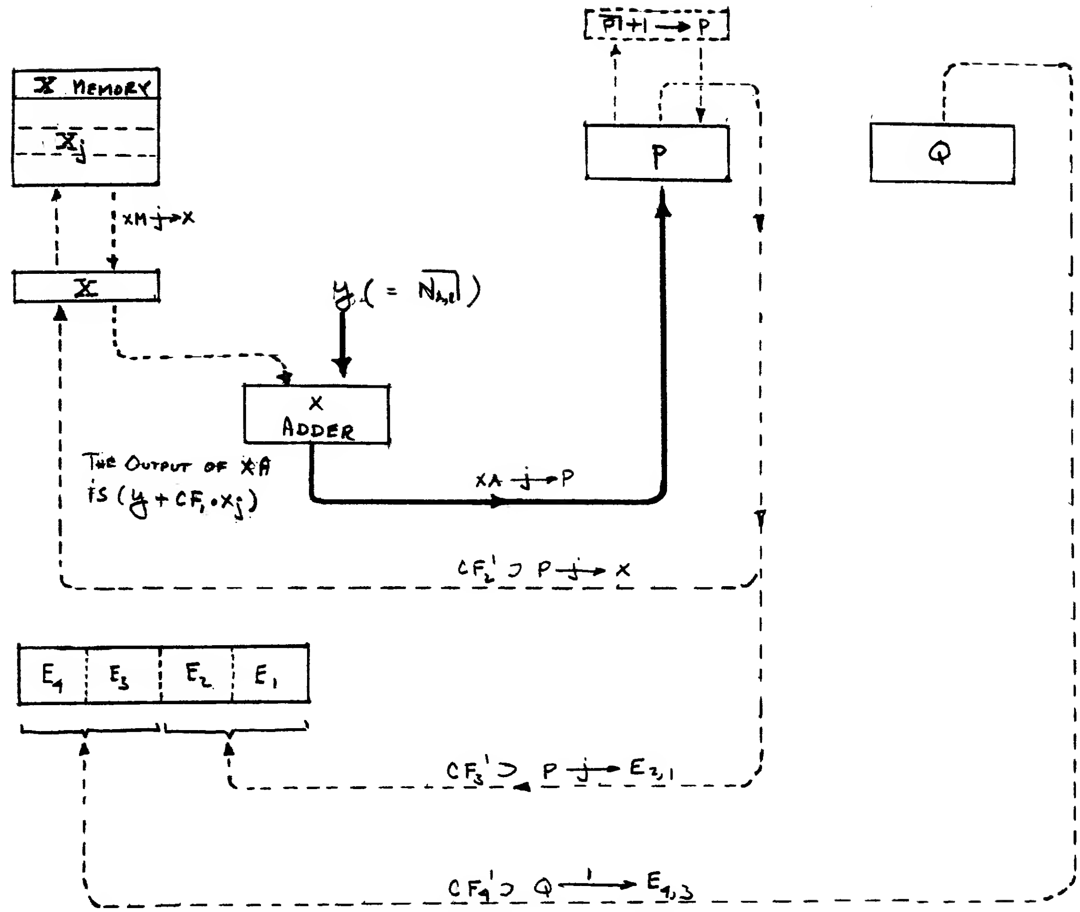
CF_2. The time that the X Memory write cycle occurs, which determines the final content of X, is conditional on CF_2. CF_2^0 starts the XWK counter in $PK^{14\alpha}$. This results in the original x_j being rewritten in X_j. When CF_2^1, the content of P is transferred into X and the XWK counter is started at $PK^{31\alpha}$. This results in the original content of P (after being indexed by one) being written in X_j.

CF_3^1. The content of P (after being indexed by one) is placed in $E_{2,1}$.

CF_4^1. The content of Q is placed in $E_{4,3}$. (Q contains the address of the operand or the last deferred address used in the instruction preceding the JMP.)

CF_5^1. The $PKIR^{DIS\ REQ}$ level is generated. The flag of the current sequence is lowered at PK^{25} and either PI_3 or CSK_4 is set in PK^{31} if $PKIR_H^0$. Note that PI_3 can also, be set redundantly, in PK^{24}.

Note that the content of P is not changed if there is an unsuppressed alarm (AL) and the Auto Start After Alarm switch is not turned on. Also, the content of X_j is not changed if there is an X parity alarm ($XPAL^1$) and the alarm is not suppressed.



CF BIT SIGNIFICANCE IN JMP									
PK	OPERATIONS INDEPENDENT OF $PKIR_{CF}$	OPERATIONS DEPENDENT ON $PKIR_{CF}$							
		CF_4		CF_3		CF_2		CF_1	
		0	1	0	1	0	1	0	1
13d	$XM \xrightarrow{j} X$								
14d						START YWK		0 XAS	1 XAS
25d			$0 \rightarrow E_{4,3}$						
31d	$XA \xrightarrow{j} P$		$Q \xrightarrow{1} E_{4,3}$		$P \xrightarrow{j} E_{2,1}$		START XWK $P \xrightarrow{j} X$		

JUMP

	24	α			
PK	25	α			$\overline{PK'} + 1 \longrightarrow PK$
			$EB^0 + (PKIR_{cf_3^0} \cdot PKIR_{cf_4^0})$ . . .	$\supset$	$\xrightarrow{31} PK$
			$EB^0 \cdot PKIR_{cf_4^1}$	$\supset$	$\xrightarrow{0} E_{4,3}$
			$PKIR_h^0 \cdot PKIR^{dis.\ req} \cdot \overline{KD^{00}}$	$\supset$	$\xrightarrow[KD]{dismiss} Flag$
	31	α	$PI^{ch\ seq}$	$\supset$	$\xrightarrow{1} PI_3$
			$PKIR^{XM}$	$\supset$	$\xrightarrow{start} XWK$
			$PKIR_h^0 \cdot PKIR^{dis.\ req} \cdot \overline{SS^{att\ req}}$ . . .	$\supset$	$\xrightarrow{1} CSK_4$
			$PKIR_{cf_2^1} \cdot (XPAL_{sup} + XPAL^0)$ . .	$\supset$	$P \xrightarrow{j} X$
			$PKIR_{cf_3^1}$	$\supset$	$P \xrightarrow{j} E_{3,1}$
			$PKIR_{cf_3^1} \cdot (\overline{AL} + AUTO\text{-}START)$ . .	$\supset$	$XA \xrightarrow{j} P$
			$PKIR_{cf_4^1}$	$\supset$	$Q \xrightarrow{1} E_{4,3}$

OP Class Decoder Lines Up:

$PKIR_{cf_1^1} \supset PKIR^{ind}$

$PKIR_{cf_2^1} \supset PKIR^{XM}$

$PKIR_{cf_5^1} \supset PKIR^{dis\ req}$, $PKIR^{def}$, $PKIR^{dis}$

JUMP ON POSITIVE INDEX

OP CODE DESCRIPTION. JPX performs a jump to the memory address specified by the base address, if the content of the index (X_j) register is positive and non-zero. The signed four bit number represented by the CF bits is then added to the index register and the result stored in X_j. JPX is a non-configurable and non-indexable instruction.

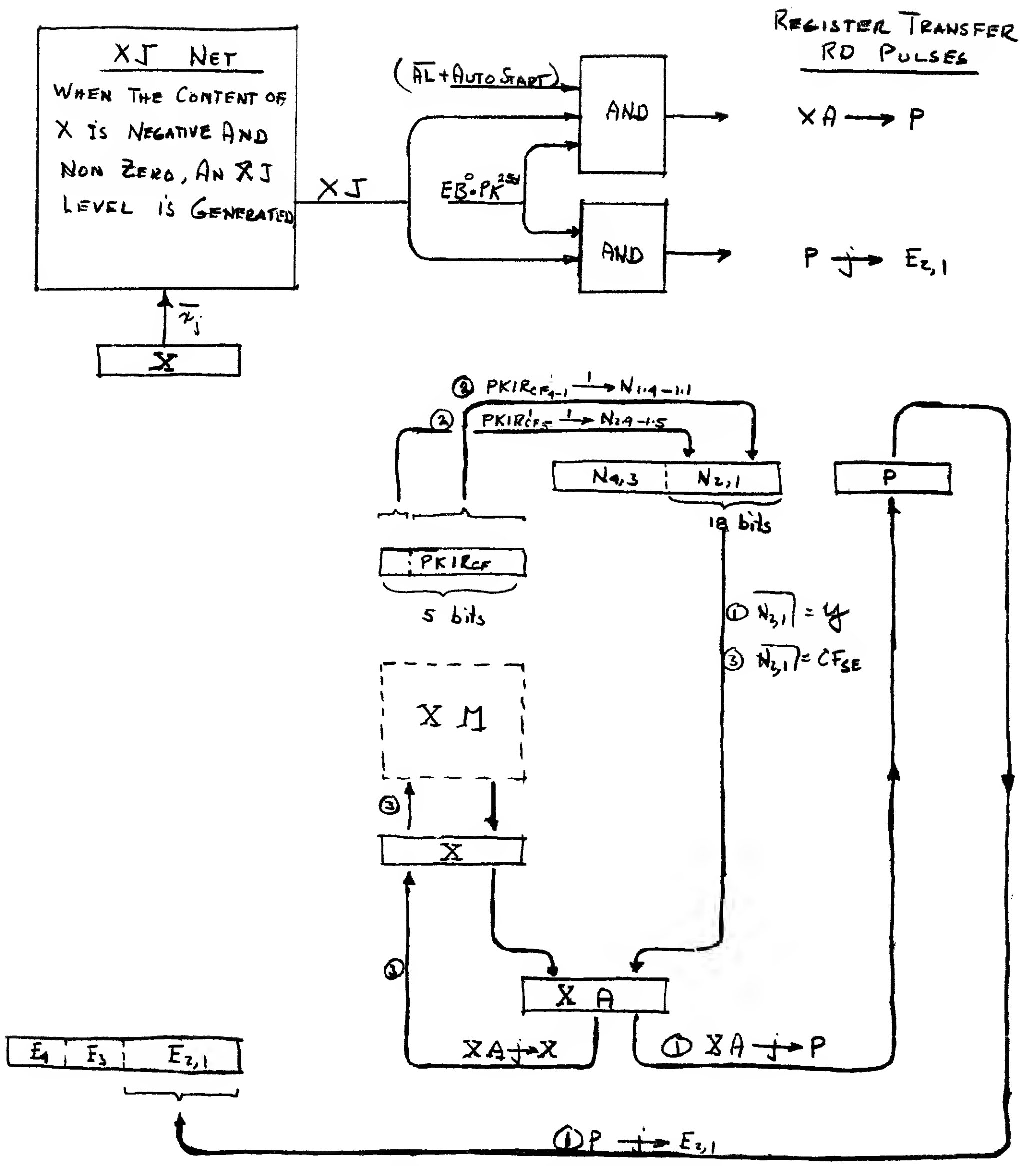
SPECIAL FEATURES. JPX has an extended PK cycle (PK^{25} through PK^{31}) and no QK cycle. The content of X is sampled by the XJ net. The XJ level is generated only if the signed content of X_j is positive and non-zero. The sign bit (CF_5) of the CF bits is extended so that an 18 bit number is formed from the 5 bit content of $PKIR_{CF}$.

DETAILS. The content of X is complemented at $PK^{15\alpha}$ and then sampled by the XJ net in $PK^{25\alpha}$. If an XJ level is generated, the content of $N_{2,1}$ is transferred into P via the X Adder. X is again complemented, restoring the original content of the index register.

The content of $PKIR_{CF}$ is then transferred into $N_{2,1}$. $N_{2,1}$ is filled up by extending the sign of the CF bits. The X Adder forms the sum of CF_{SE} and the index. This sum is then transferred to X and written in the selected X Memory register.

The $PKIR^{DIS\ REQ}$ level is generated if the XJ level is generated. XJ has meaning only until PK^{25}. Hence the change sequence conditions are examined at PK^{24}, the wait conditions at PK^{25}, and the flag of the current sequence lowered at PK^{25} before PK leaves PK^{25}.

As in JMP, the content of P and the content of X_j are not changed if the parity alarms occur.



JPX ILLUSTRATIVE EXAMPLE (ASSUME $PKIR_{CF_S} = 1$ AND XJ IS GENERATED)								
PK	STEP	$N_{2,1}$	XA	X	P	$E_{2,1}$	$PKIR_{CF}$	OPERATION
—	—	y	y	x_j	p	$e_{2,1}$	CF	
252	①	0	0	x_j	y	p		JUMP
262	②	CF_{SE}	CF_{SE}	x_j		p		SIGN EXTENSION
312	③	CF_{SE}	$CF_{SE} \overset{*}{+} x_j$	$CF_{SE} \overset{*}{+} x_j$		p		MODIFY INDEX

* THE + SIGNS ON THIS CHART INDICATE AN ARITHMETIC SUM NOT A LOGICAL SUM

JUMP ON POSITIVE INDEX

	Time	Phase	Condition		Action
	24	α			
	25	α	EB°	$\supset$	$\lfloor c \rightarrow X$
			$EB^{\circ} + \overline{XJ}$	$\supset$	$\lfloor 0 \rightarrow N_{2,1}$, $\lfloor 1 \rightarrow XAC$
			$EB^{\circ} \cdot XJ$	$\supset$	$P \xrightarrow{j} E_{2,1}$
			$EB^{\circ} \cdot XJ \cdot (\overline{AL} + \text{AUTO-START})$	$\supset$	$XA \xrightarrow{j} P$
			EB^{1}	$\supset$	$\overline{PK'} + 1 \xrightarrow{/} PK$
			$PKIR^{\circ}_{h} \cdot PKIR^{dis\ req} \cdot \overline{SS^{att\ req}}$	$\supset$	$\lfloor 1 \rightarrow CSK_4$
			$PKIR^{\circ}_{h} \cdot PKIR^{dis\ req} \cdot \overline{KD^{00}}$	$\supset$	$\lfloor \frac{dismiss}{KD} \rightarrow$ Flag
PK	26	α			$\lfloor 1 \rightarrow XAS$
					$PKIRcf_{4-1} \xrightarrow{1} N_{1.4-1.1}$
			$PKIRcf^{1}_{5}$	$\supset$	$\lfloor 1 \rightarrow N_{2.9-1.5}$
	27	α			
	28	α			$\lfloor 31 \rightarrow PK$
	31	α	$XPAL_{sup} + XPAL^{\circ}$	$\supset$	$XA \xrightarrow{j} X$
			$PKIR^{XM}$	$\supset$	$\lfloor start \rightarrow XWK$

OP Class Decoder Lines Up: $XJ \supset$ $PKIR^{dis\ req}$
$PKIR^{dis}$
$PKIR^{def}$
$PKIR^{XM}$
$PKIR^{jx}$

$$XJ = X^{1}_{2.9} \cdot (X^{0}_{2.8} + \cdots + X^{0}_{1.1})$$

This instruction jumps if the index is positive & non zero.
It then adds the signed 4 bit number represented by the PKIRcf bits to the index register.

C.A.N 7-11-61

JUMP ON NEGATIVE INDEX

OP CODE DESCRIPTION. JNX performs a jump to the memory address specified by the base address, if the content of the index (X_j) register is negative and non-zero. The signed four bit number represented by the CF bits is then added to the index register and the result stored in X_j. JNX is a non-configurable and non-indexable instruction.

DETAILS. (The execution logic for JNX is identical to that for JPX, except that the complement X pulses generated at $PK^{15\alpha}$ and $PK^{25\alpha}$ in JPX do not occur in JNX. In JNX, the XJ level occurs if the content of X_j is negative and non-zero. See JPX description.)

JUMP ON NEGATIVE INDEX

24	α			
25	α	$EB^{o} + \overline{XJ}$	$\supset$	$\lfloor 0 \rightarrow N_{2,1}$, $\lfloor 1 \rightarrow XAC$
		$EB^{o} \cdot XJ$	$\supset$	$P \xrightarrow{j} E_{2,1}$
		$EB^{o} \cdot XJ \cdot (\overline{AL} + \text{AUTO-START})$	$\supset$	$XA \xrightarrow{j} P$
		EB^{1}	$\supset$	$\overline{PK'} + 1 \xrightarrow{\ } PK$
		$PKIR^{o}_{h} \cdot PKIR^{dis\ req} \cdot \overline{SS^{att\ req}}$	$\supset$	$\lfloor 1 \rightarrow CSK_4$
		$PKIR^{o}_{h} \cdot PKIR^{dis\ req} \cdot \overline{KD^{oo}}$	$\supset$	$\lfloor dismiss \rightarrow Flag$
PK				
26	α			$\lfloor 1 \rightarrow XAS$
				$PKIRcf_{4-1} \xrightarrow{1} N_{1.4-1.1}$
		$PKIR_{cf}{}^{1}_{5}$	$\supset$	$\lfloor 1 \rightarrow N_{2.9-1.5}$
27	α			
28	α			$\lfloor 31 \rightarrow PK$
31	α	$XPAL_{sup} + XPAL^{o}$	$\supset$	$XA \xrightarrow{j} X$
		$PKIR^{XM}$	$\supset$	$\lfloor start \rightarrow XWK$

OP Class Decoder Lines Up: $XJ \supset$ $PKIR^{dis\ req}$
$PKIR^{dis}$
$PKIR^{def}$
$PKIR^{XM}$
$PKIR^{jx}$

$$XJ = X^{1}_{2.9} \cdot (X^{o}_{2.8} + \cdots\cdots + X^{o}_{1.1})$$

This instruction jumps if the index is negative & non zero.
It then adds the signed 4 bit number represented by the $PKIR_{cf}$ bits to the index register.

C.AN 7-11-61

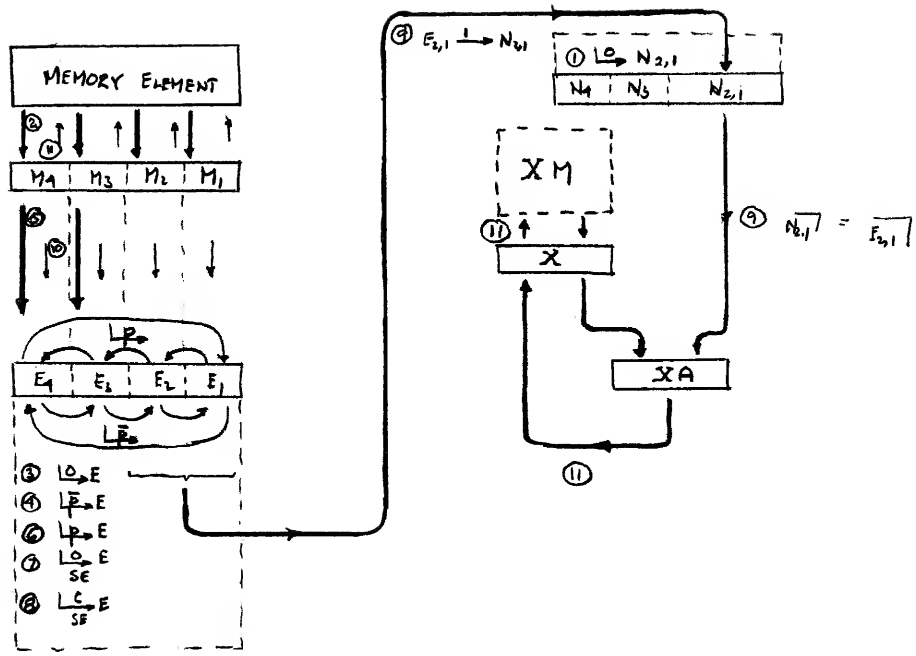
AUGMENT INDEX (FROM MEMORY)

OP CODE DESCRIPTION. AUX "augments" the content of the specified index register (X_j) with part of the content of the selected Memory Element register. The sum is stored in the specified index register. AUX is a configurable, but non-indexable instruction.

SPECIAL FEATURES. The addition that occurs in the X Adder treats the contents of $N_{2,1}$ and X as two 18 bit signed numbers.

DETAILS. The E register is cleared. After the normal configuration and sign extension process that takes place in a load type instruction has occurred, the content of $E_{2,1}$ is algebraically added to the content of X. The result of this addition is then placed in X and the content of X written in the X Memory.

Note that content of X_j is not changed if an unsuppressed X parity alarm occurred.



AUX ILLUSTRATIVE EXAMPLE (ASSUME $x_{j4}=1$, $y_{j4}=1$, $y_{4.9}=1$ AND)

QK	STEP	MEMORY	M				E				$N_{2,1}$		XA		X		XM		OPERATION
—	—	y	m_4	m_3	m_2	m_1	e_4	e_3	e_2	e_1	—	-	—	-	x_{j2}	x_{j1}			
01α	①										0	0							CLEAR $N_{2,1}$
02-11	②		y_4	y_3	y_2	y_1													READ
10α	③						0	0	0	0									CLEAR E
11β	④						0	0	0	0									CONFIGURATION
13α	⑤						y_4	y_3	0	0									
13β	⑥						y_3	0	0	y_4									
14α	⑦						y_3	0	0	y_4									SIGN EXTENSION
14β	⑧						y_3	1	1	y_4									
21α	⑨										1	y_4	$1+x_{j2}$*	y_4+x_{j2}*	x_{j2}	x_{j1}			LOAD $N_{2,1}$ WITH $\overline{E_{2,1}}$
23α	⑩						y_4	y_3	y_2	y_1									ULTIMATE PULSE
21-31	⑪	y_4 y_3 y_2 y_1													$1+x_{j2}$*	y_4+x_{j2}*	$1+x_{j2}$	y_4+x_{j2}*	REWRITE

* THE + SIGNS ON THIS CHART INDICATE AN 18 bit ARITHMETIC SUM AND NOT A LOGICAL SUM

AUGMENT INDEX (FROM MEMORY)

PK	24	α		$\lfloor \xrightarrow{00} PK$
	00	α	QI^{start} . . . . $\supset$	$\lfloor \xrightarrow{start} \diamond FK$
	01	α		$\lfloor \xrightarrow{0} N_{2,1}$ $\lfloor \xrightarrow{1} XAC$
			SEE QKM TIMING	
	09	α		
	10	α		$\lfloor \xrightarrow{0} E$ $\lfloor \xrightarrow{1} XAS$
	11	α		$\lfloor \xrightarrow{13} QK$
		β		$\lfloor \xrightarrow{\bar{P}} E$
QK	13	α		$\lfloor \xrightarrow{1} XR$ $\lfloor \xrightarrow{1} XB$ $M \xrightarrow[\alpha\, p]{0,1} E$
		β		$\lfloor \xrightarrow{P} E$
	14	α		$\lfloor \xrightarrow[SE]{0} E$ $\lfloor \xrightarrow{0} PI_1$ $\lfloor \xrightarrow{1} XAC$ $\lfloor \xrightarrow{21} QK$
		β		$\lfloor \xrightarrow[SE]{C} E$
	21	α		$E_{2,1} \xrightarrow{1} N_{2,1}$
	22	α		
	23	α		$\lfloor \xrightarrow{0} EB$ $M \xrightarrow{0,1} E$
			SEE QKM TIMING	
	31	α	$XPAL_{sup}$ + $XPAL^{o}$ . . . . . $\supset$	$\lfloor \xrightarrow{start} XWK$ $XA \xrightarrow{j} X$

OP Class Decoder Lines Up:

$PKIR^{def}$
$PKIR^{QK}$
$QKIR^{ld}$
$QKIR^{load}$

CAN 7-11-61

RESET INDEX (FROM MEMORY)

OP CODE DESCRIPTION. RSX "resets" the content of the specified index register (X_j) with part of the content of the selected Memory Element register. RSX is a configurable, but non-indexable instruction.

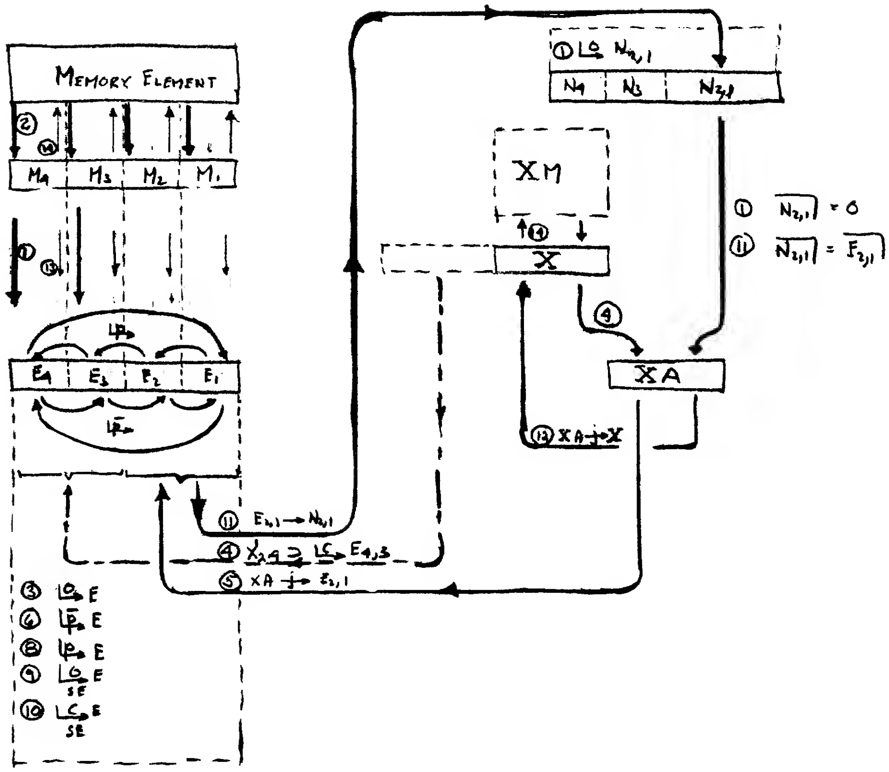
SPECIAL FEATURES. The content of the X register is treated as an 18 bit signed word. By extending the sign bit of this word to the left, a 36 bit word is formed. This 36 bit word then enters into the configuration process just as the 36 bit content of A would enter into the configuration process in a LDA.

DETAILS. After E is cleared, quarter 3 and 4 of E are complemented based on the sign of the word in X. The content of X is then jammed into $E_{2,1}$. Note that the effect is as if X contained a 3rd and 4th quarter and the sign of X were extended into these quarters.

After E is loaded with the sign extended content of X, the normal configuration and sign extension process that takes place in a load type instruction occurs.

The content of $E_{2,1}$ is then routed through $N_{2,1}$ and XA into X. The content of X is then written in the X Memory by the XWK counter.

Note that the content of X_j is not changed if an unsuppressed X parity alarm occurred.



RSX ILLUSTRATIVE EXAMPLE (Assume $x_{2,4}=1$, $y_{3,4}=1$, $y_{4,4}=1$ And)																			
QK	STEP	MEMORY	M				E				$N_{2,1}$		XA		X		XM		OPERATION
—	—	y	m_4	m_3	m_2	m_1	e_4	e_3	e_2	e_1	—	—	—	—	x_{j2}	x_{j1}			
01α	①										0	0							Clear $N_{2,1}$
02-11	②		y_4	y_3	y_2	y_1													Read
10α	③						0	0	0	0									Clear E
10β	④						1	1	0	0			x_{j2}	x_{j1}					Extend Sign of x_j
11α	⑤						1	1	x_{j2}	x_{j1}									Load $E_{2,1}$ with $\overline{XA}$
11β	⑥						x_{j1}	1	1	x_{j2}									Configuration
13α	⑦						y_4	y_3	1	x_{j2}									
13β	⑧						y_3	1	x_{j2}	y_4									
14α	⑨						y_3	0	0	y_4									Sign Extension
14β	⑩						y_3	1	1	y_4									
21α	⑪										1	y_4	1	y_4					Load $N_{2,1}$ with $\overline{E_{2,1}}$
22α	⑫														1	y_4			Load X with $\overline{XA}$
23	⑬						y_4	y_3	y_2	y_1									Ultimate Pulse
21-31	⑭	y															1	y_4	Rewrite

RESET INDEX (FROM MEMORY)

Register	Time	Phase	Condition	Action
PK	24	α		$\lfloor 00 \rightarrow PK$
	00	α	QI^{start} . . . . ⊃	$\lfloor start \rightarrow FK$
	01	α		$\lfloor 0 \rightarrow N_{2,1}$ $\lfloor 1 \rightarrow XAC$
	SEE QKM TIMING			
	09	α		
	10	α		$\lfloor 0 \rightarrow E$ $\lfloor 1 \rightarrow XAS$
		β	$X^1_{2\cdot9}$ . . . . ⊃	$\lfloor C \rightarrow E_{4,3}$
	11	α		$XA \xrightarrow{1} E_{2,1}$ $\lfloor 13 \rightarrow QK$
		β		$\lfloor \bar{P} \rightarrow E$
QK	13	α		$\lfloor 1 \rightarrow XR$ $\lfloor 1 \rightarrow XB$ $M \xrightarrow[\alpha,P]{0,1} E$
		β		$\lfloor P \rightarrow E$
	14	α		$\lfloor 0_{SE} \rightarrow E$ $\lfloor 0 \rightarrow PI_1$ $\lfloor 21 \rightarrow QK$
		β		$\lfloor C_{SE} \rightarrow E$
	21	α		$E_{2,1} \xrightarrow{1} N_{2,1}$ $\lfloor 0 \rightarrow XAS$
	22	α	$XPAL_{sup}$ + $XPAL^{\circ}$ . . . . . ⊃	$\lfloor start \rightarrow XWK$ $XA \xrightarrow{j} X$
	23	α		$M \xrightarrow{0,1} E$ $\lfloor 0 \rightarrow EB$
	SEE QKM TIMING			
	31	α		

OP Class Decoder Lines Up: $PKIR^{def}$, $PKIR^{QK}$, $QKIR^{load}$, $QKIR^{ld}$ & $QKIR^{X}$

C.A.N 7-11-61

OP CODE DESCRIPTION. SKX performs a conditional skip based on a comparison of the content of the specified index register (X_j) and the base address, or it replaces or augments the content of the specified index register with the positive or negative value of the base address. SKX can also lower the flag of its own sequence and raise the flag of any other sequence. SKX is a non-configurable and non-indexable instruction.

SPECIAL FEATURES. SKX has an extended PK cycle (PK^{25} through PK^{31}) and no QK cycle. The $PKIR_{CF}$ bits are used for a special purpose.

DETAILS. The instruction may be differentiated into two distinct types based on $PKIR_{CF_3}$. If $PKIR^0_{CF_3}$, the instruction is a $\overline{\text{SKIP}}$ type and if $PKIR^1_{CF_3}$, the instruction is a SKIP type. There are four versions of each type, based on the state of the $PKIR_{CF_2}$ and $PKIR_{CF_1}$ bits.

Consider the $\overline{\text{SKIP}}$ example illustrated. The content of the index register is loaded into X at $PK^{13\alpha}$. The content of X is then replaced by jamming the base address via the X Adder into X. The content of X is then written into the X Memory by XWK.

The variations in the $\overline{\text{SKIP}}$ instructions involve changing the sign of the base address (CF^1_1) and/or adding the original content of the selected X register (CF^1_2) before storing it in the X Memory.

Similarly, there are four SKIP type instructions. Consider the SKIP example illustrated. The content of the index register is loaded into X at $PK^{13\alpha}$. The base address is then indexed and loaded into X where the sum is complemented. Note that the content of X will be negative and non-zero only when the negative value of the base address is arithmetically less than the value of the content of the index register.

The content of X is now sampled by the XJ net. If the content of X is in fact negative and non-zero, the XJ level is generated and P is indexed. Note that P was previously indexed in $PK^{24\alpha}$. The balance of the instruction restores the original value of the index register in X. The content of X is then written in the X Memory by the XWK counter.

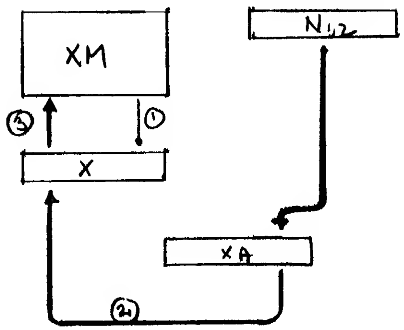
The variations in the SKIP instructions involve the type of comparison of the base address and the index register that is used to make the SKIP decision. The comparisons involve the base address or its complement (CF_1) and either an in equality or greater-than-less than (CF_2) comparison.

The logic of all eight variations are listed separately on the timing chart.

SKX also has flag raising and lowering features. When the $PKIR^{DIS\ REQ}$ level is generated (CF_5^1), the flag of the current sequence is lowered. If $PKIR_{CF_4}^1$, the flag of the sequence specified by the J bits is raised. Note that if $CF_5^1 \cdot CF_4^1 \cdot K^{eq}\ J$, i.e., the current sequence is both dismissed and has its flag raised, then the $PKIR^{DIS\ REQ}$ level is not generated. The flag in this case is lowered in PK^{25} and then raised in PK^{26}. The sequence change condition are examined in both PK^{24} and PK^{31} and hence PI_3 can be set in both these states. The instruction cannot lower flags of other sequences, so that the examination in PK^{24} cannot erroneously set PI_3. The wait conditions are examined only in PK^{31}.

SKX ILLUSTRATIVE EXAMPLES

$\overline{\text{SKIP}}$ $\quad CF_3^0 \cdot CF_2^0 \cdot CF_1^0 \supset n \longrightarrow X_j$

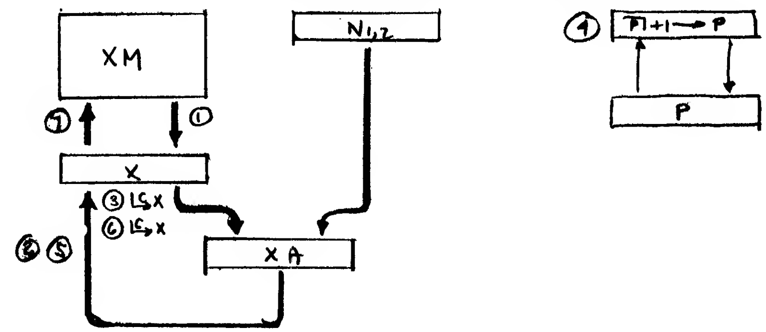


PK	STEP	XM	X	XA	OPERATION
—	—	x_j	—	n	
13½	①		x_j	n	READ
26½	②		n	n	LOAD X

$PK^{31½} \supset$ LSTART XWK

XWK					
00-07	③	n	n	n	REWRITE

SKIP $\quad CF_3^1 \cdot CF_2^1 \cdot CF_1^1 \supset (-n < x_j \supset \text{SKIP})$



PK	STEP	XM	X	XA	OPERATION
—	—	x_j	—	n	
13½	①		x_j	↓	READ
26½	②		$n+x_j$		INDEX
27½	③		$-(n+x_j)$		COMPLEMENT
28½	④				$\overline{XI}$ IS SAMPLED BY XJ NET. IF XI IS NEGATIVE & $\overline{\text{ZERO}}$, XJ IS GENERATED & SKIP OCCURS
30	⑤		$-x_j$		$n-(n+x_j)=-x_j$
31	⑥		x_j	↓	COMPLEMENT

$PK^{31½} \supset$ LSTART → XWK

XWK					
00-07	⑦	x_j	x_j	n	REWRITE

SKIP ON INDEX

		SKIP				$\overline{SKIP}$			
		$cf_3^1 \cdot cf_2^1 \cdot cf_1^1$	$cf_3^1 \cdot cf_2^1 \cdot cf_1^0$	$cf_3^1 \cdot cf_2^0 \cdot cf_1^1$	$cf_3^1 \cdot cf_2^0 \cdot cf_1^0$	$cf_3^0 \cdot cf_2^1 \cdot cf_1^1$	$cf_3^0 \cdot cf_2^1 \cdot cf_1^0$	$cf_3^0 \cdot cf_2^0 \cdot cf_1^1$	$cf_3^0 \cdot cf_2^0 \cdot cf_1^0$
PK		$(-r < x_j \supset SKIP)$	$(r > x_j \supset SKIP)$	$(-r \neq x_j \supset SKIP)$	$(r \neq x_j \supset SKIP)$	$(-r + x_j \longrightarrow x_j)$	$(r + x_j \longrightarrow x_j)$	$(-r \longrightarrow x_j)$	$(r \longrightarrow x_j)$
13	α	$XM \xrightarrow{j} X$	$XM \xrightarrow{j} X$	$XM \xrightarrow{j} X$	$XM \xrightarrow{j} X$	$XM \xrightarrow{j} X$	$XM \xrightarrow{j} X$	$XM \xrightarrow{j} X$	$XM \xrightarrow{j} X$
14	α	$(PI_2^0 \cdot PKIR^{ind} + PI_5^1) \rightarrow XAS$ $\lfloor 1 \rightarrow XAC$	$(PI_2^0 \cdot PKIR^{ind} + PI_5^1) \rightarrow XAS$ $\lfloor 1 \rightarrow XAC$	$(PI_2^0 \cdot PKIR^{ind} + PI_5^1) \rightarrow XAS$ $\lfloor 1 \rightarrow XAC$	$(PI_2^0 \cdot PKIR^{ind} + PI_5^1) \rightarrow XAS$ $\lfloor 1 \rightarrow XAC$	$(PI_2^0 \cdot PKIR^{ind} + PI_5^1) \rightarrow XAS$ $\lfloor 1 \rightarrow XAC$	$(PI_2^0 \cdot PKIR^{ind} + PI_5^1) \rightarrow XAS$	$(PI_2^0 \cdot PKIR^{ind} + PI_5^1) \rightarrow XAS$	$(PI_2^0 \cdot PKIR^{ind} + PI_5^1) \rightarrow XAS$
15	α		$PI_2^0 \supset \lfloor c \rightarrow X$		$PI_2^0 \supset \lfloor c \rightarrow X$	$PI_2^0 \supset \lfloor c \rightarrow X$	$\lfloor 1 \rightarrow XAC$	$\lfloor 1 \rightarrow XAC$	$\lfloor 1 \rightarrow XAC$
SEE PKM TIMING									
25	α	$(PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{KD^{00}}) \supset \lfloor dismiss_{KD} \rightarrow Flag$	$(PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{KD^{00}}) \supset \lfloor dismiss_{KD} \rightarrow Flag$	$(PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{KD^{00}}) \supset \lfloor dismiss_{KD} \rightarrow Flag$	$(PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{KD^{00}}) \supset \lfloor dismiss_{KD} \rightarrow Flag$	$(PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{KD^{00}}) \supset \lfloor dismiss_{KD} \rightarrow Flag$	$(PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{KD^{00}}) \supset \lfloor dismiss_{KD} \rightarrow Flag$	$(PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{KD^{00}}) \supset \lfloor dismiss_{KD} \rightarrow Flag$	$(PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{KD^{00}}) \supset \lfloor dismiss_{KD} \rightarrow Flag$
26	α	$\lfloor 1 \rightarrow XAC$ $XPAL_{sup} + XPAL^0 \supset XA \xrightarrow{j} X$ $PKIR_{cf_4^1} \supset \lfloor 1_{ND} \rightarrow Flag$	$\lfloor 1 \rightarrow XAC$ $XPAL_{sup} + XPAL^0 \supset XA \xrightarrow{j} X$ $PKIR_{cf_4^1} \supset \lfloor 1_{ND} \rightarrow Flag$	$\lfloor 1 \rightarrow XAC$ $XPAL_{sup} + XPAL^0 \supset XA \xrightarrow{j} X$ $PKIR_{cf_4^1} \supset \lfloor 1_{ND} \rightarrow Flag$	$\lfloor 1 \rightarrow XAC$ $XPAL_{sup} + XPAL^0 \supset XA \xrightarrow{j} X$ $PKIR_{cf_4^1} \supset \lfloor 1_{ND} \rightarrow Flag$	$\lfloor 1 \rightarrow XAC$ $XPAL_{sup} + XPAL^0 \supset XA \xrightarrow{j} X$ $PKIR_{cf_4^1} \supset \lfloor 1_{ND} \rightarrow Flag$	$\lfloor 1 \rightarrow XAC$ $XPAL_{sup} + XPAL^0 \supset XA \xrightarrow{j} X$ $PKIR_{cf_4^1} \supset \lfloor 1_{ND} \rightarrow Flag$	$\lfloor 1 \rightarrow XAC$ $XPAL_{sup} + XPAL^0 \supset XA \xrightarrow{j} X$ $PKIR_{cf_4^1} \supset \lfloor 1_{ND} \rightarrow Flag$	$\lfloor 1 \rightarrow XAC$ $XPAL_{sup} + XPAL^0 \supset XA \xrightarrow{j} X$ $PKIR_{cf_4^1} \supset \lfloor 1_{ND} \rightarrow Flag$
27	α	$\lfloor c \rightarrow X$	$\lfloor c \rightarrow X$	$\lfloor c \rightarrow X$ $XJ \supset \overline{P}+1 \rightarrow P$	$\lfloor c \rightarrow X$ $XJ \supset \overline{P}+1 \rightarrow P$				
28	α	$XJ \supset \overline{P}+1 \rightarrow P$	$XJ \supset \overline{P}+1 \rightarrow P$	$XJ \supset \overline{P}+1 \rightarrow P$	$XJ \supset \overline{P}+1 \rightarrow P$				
29	α					$\lfloor 31 \rightarrow PK$	$\lfloor 31 \rightarrow PK$	$\lfloor 31 \rightarrow PK$	$\lfloor 31 \rightarrow PK$
30	α	$XPAL_{sup} + XPAL^0 \supset XA \xrightarrow{j} X$	$XPAL_{sup} + XPAL^0 \supset XA \xrightarrow{j} X$	$XPAL_{sup} + XPAL^0 \supset XA \xrightarrow{j} X$	$XPAL_{sup} + XPAL^0 \supset XA \xrightarrow{j} X$				
31	α	$\lfloor c \rightarrow X$ $\lfloor start \rightarrow XWK$ $PI^{ch\ seq} \supset \lfloor 1 \rightarrow PI_3$ $(PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{SS^{att\ req}}) \supset \lfloor 1 \rightarrow CSK_4$	$\lfloor start \rightarrow XWK$ $PI^{ch\ seq} \supset \lfloor 1 \rightarrow PI_3$ $(PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{SS^{att\ req}}) \supset \lfloor 1 \rightarrow CSK_4$	$\lfloor c \rightarrow X$ $\lfloor start \rightarrow XWK$ $PI^{ch\ req} \supset \lfloor 1 \rightarrow PI_3$ $(PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{SS^{att\ req}}) \supset \lfloor 1 \rightarrow CSK_4$	$\lfloor start \rightarrow XWK$ $PI^{ch\ req} \supset \lfloor 1 \rightarrow PI_3$ $(PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{SS^{att\ req}}) \supset \lfloor 1 \rightarrow CSK_4$	$\lfloor c \rightarrow X$ $\lfloor start \rightarrow XWK$ $PI^{ch\ req} \supset \lfloor 1 \rightarrow PI_3$ $(PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{SS^{att\ req}}) \supset \lfloor 1 \rightarrow CSK_4$	$\lfloor start \rightarrow XWK$ $PI^{ch\ req} \supset \lfloor 1 \rightarrow PI_3$ $(PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{SS^{att\ req}}) \supset \lfloor 1 \rightarrow CSK_4$	$\lfloor c \rightarrow X$ $\lfloor start \rightarrow XWK$ $PI^{ch\ req} \supset \lfloor 1 \rightarrow PI_3$ $(PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{SS^{att\ req}}) \supset \lfloor 1 \rightarrow CSK_4$	$\lfloor start \rightarrow XWK$ $PI^{ch\ req} \supset \lfloor 1 \rightarrow PI_3$ $(PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{SS^{att\ req}}) \supset \lfloor 1 \rightarrow CSK_4$

OP Class Decoder Lines Up:

$PKIR_{cf_2^1} + PKIR_{cf_3^1} \cdots\cdots \supset PKIR^{ind}$

$PKIR_{cf_5^1} \cdot (\overline{K^{eqJ} \cdot PKIR_{cf_4^1}}) \cdot\cdot \supset PKIR^{dis\ req}$

$PKIR^{XM}$

$PKIR^{def}$

$PKIR^{dis}$

$$XJ = X_{2.9}^1 \cdot (X_{2.8}^0 + \cdots\cdots + X_{1.1}^0)$$

r = (last) direct address.

Ordinarily, if no deferred address, then $r = y$

CAN 6-13-61

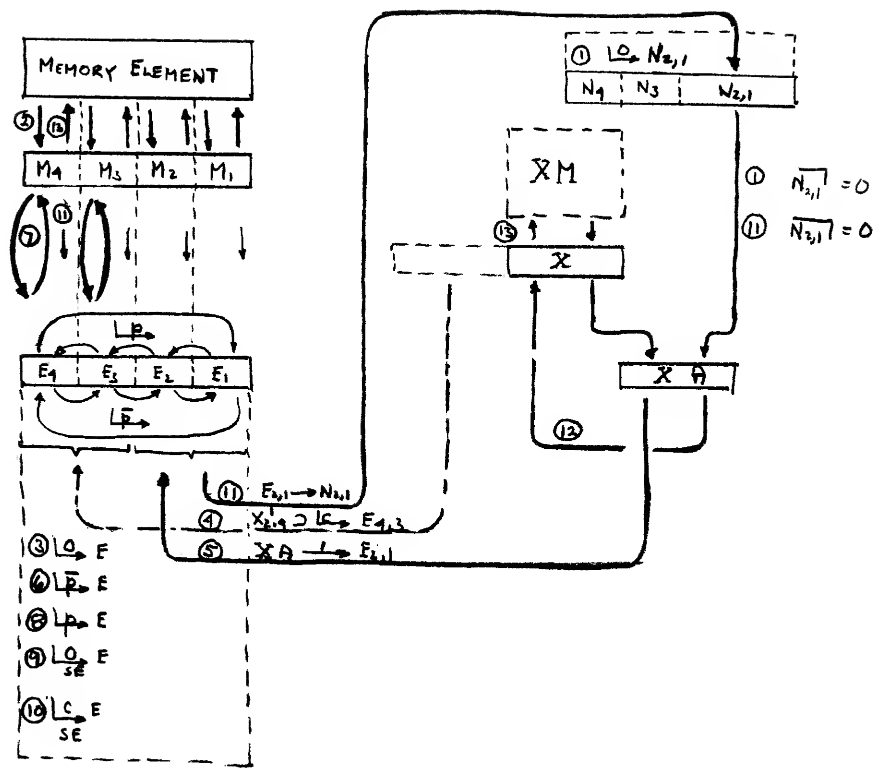
EXCHANGE INDEX (WITH MEMORY)

OP CODE DESCRIPTION. EXX "exchanges" the content of the specified index register (X_j) with part of the content of the selected Memory Element register. EXX is a configurable, but non-indexable instruction.

SPECIAL FEATURES. The content of the X register is treated as an 18 bit signed word. By extending the sign of this word, a 36 bit word is formed. This 36 bit word then enters into the configuration process just as the content of A would enter into the configuration process in an EXA.

DETAILS. E is cleared and quarters 3 and 4 of E are complemented if the sign of the word in X is a ONE. The content of X is then jammed into $E_{2,1}$. The effect is as if X contained a 3rd and 4th quarter and the sign of X were extended into these quarters.

After E is loaded with the content of X, the content of M and E are exchanged under configuration and sign extension control. The content of M is then stored in the selected Memory Element register and the content of $E_{2,1}$ is placed via $N_{2,1}$ and XA in X. The content of X is then written in the X Memory by the XWK counter.



EXX ILLUSTRATIVE EXAMPLE (ASSUME $x_{2.4}=1$, $y_{3.4}=1$, $y_{4.4}=1$ AND)

QK	STEP	MEMORY	M				E				$N_{2,1}$		X A		X		X M	OPERATION
—	—	y	m_4	m_3	m_2	m_1	e_4	e_3	e_2	e_1			—	—	x_{j2}	x_{j1}		
01α	①										0	0						CLEAR $N_{2,1}$
02-11	②		y_4	y_3	y_2	y_1												READ
10α	③						0	0	0	0								CLEAR E
10β	④						1	1	0	0			x_{j2}	x_{j1}				EXTEND SIGN x_j
11α	⑤						1	1	x_{j2}	x_{j1}								LOAD $E_{2,1}$ WITH $\overline{XA}$
11β	⑥						x_{j1}	1	1	x_{j2}								CONFIGURATION
13α	⑦		x_{j1}	1	y_2	y_1	y_4	y_3	1	x_{j2}								
13β	⑧						y_3	1	x_{j2}	y_4								
14α	⑨						y_3	0	0	y_4								SIGN EXTENSION
14β	⑩						y_3	1	1	y_4								
21α	⑪						x_{j1}	1	y_2	y_1	1	y_4	1	y_4				LOAD $N_{2,1}$ WITH $\overline{E_{2,1}}$
22α	⑫														1	y_4		LOAD X WITH $\overline{XA}$
21-31	⑬	x_{j1} 1 y_2 y_1															1 y_4	REWRITE

EXCHANGE INDEX (WITH MEMORY)

PK	24	α		$L_{00} \rightarrow PK$
	00	α	QI^{start} . . . . $\supset$	$L_{start} \rightarrow FK$
	01	α		$L_0 \rightarrow N_{2,1}$ $L_1 \rightarrow XAC$
	SEE QKM TIMING			
	09	α		
	10	α		$L_0 \rightarrow E$ $L_1 \rightarrow XAS$
		β	$X'_{2.9}$ . . . . $\supset$	$L_c \rightarrow E_{3,4}$
	11	α		$XA \xrightarrow{1} E_{2,1}$
		β		$L_{\bar{p}} \rightarrow E$
	12			
QK	13	α	$\overline{MPA}$ . . . . $\supset$	$L_1 \rightarrow XR$, $L_1 \rightarrow XB$ $M \xrightarrow[\alpha,p]{0,1} E$ $E \xrightarrow[\alpha,p]{0,1} M$
		β		$L_p \rightarrow E$
	14	α		$L_0 \rightarrow E$ (SE) $L_0 \rightarrow PI_1$ $L_{21} \rightarrow QK$
		β		$L_c \rightarrow E$ (SE)
	21	α	$\overline{QKM^{VFF}}$ . . . . $\supset$	$L_0 \rightarrow XAS$ $E_{2,1} \xrightarrow{1} N_{2,1}$ $M \xrightarrow{0,1} E$ $L_0 \rightarrow EB$
	22	α	$XPAL_{sup} + XPAL^{\circ}$ . . . . . . $\supset$	$L_{start} \rightarrow XWK$ $XA \xrightarrow{j} X$
	23	α		$L_0 \rightarrow EB$
	SEE QKM TIMING			
	31	α		

OP Class Decoder Lines Up: $PKIR^{QK}$, $QKIR^{ld}$, $QKIR^{st}$, $QKIR^{store}$, $QKIR^{X}$

CAN
7-11-61

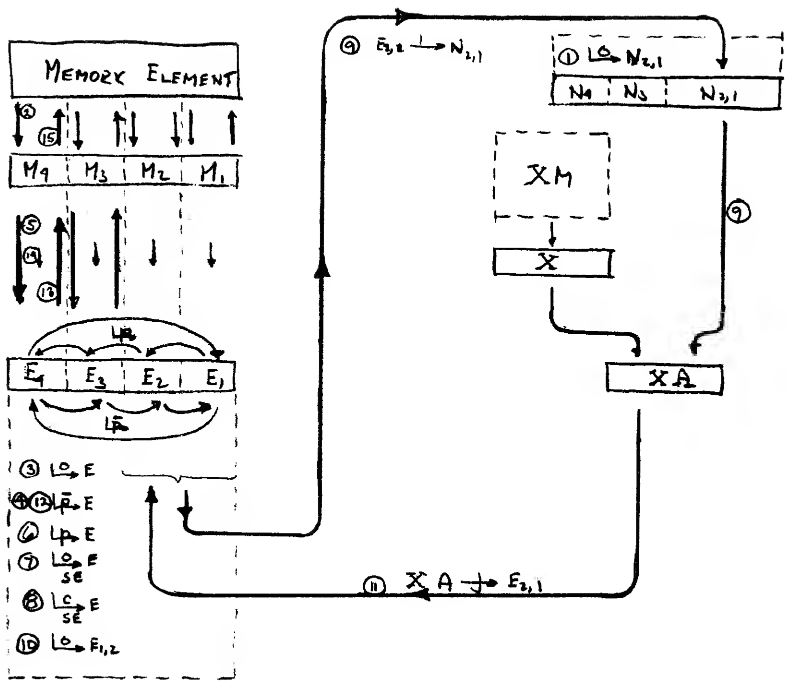
ADD INDEX (TO MEMORY)

OP CODE DESCRIPTION. ADX adds the content of the specified index register (X_j) to part of the content of the selected Memory Element register. The sum is stored back in the selected Memory Element register. ADX is a configurable, but non-indexable type instruction.

SPECIAL FEATURES. The addition that occurs in the X Adder treats the content of $N_{2,1}$ and X as two, 18 bit signed numbers. Two distinct configuration processes take place during the instruction execution logic.

DETAILS. E is cleared. Then, after the normal configuration and sign extension process that takes place in a load type instruction has occurred, the content of $E_{2,1}$ is transferred to $N_{2,1}$ and algebraically added to the content of X. The result of this addition is transferred back into $E_{2,1}$.

The normal inverse configuration process that takes place during a store type instruction then takes place. Finally the content of M is stored in the selected Memory Element register.



ADX ILLUSTRATIVE EXAMPLE (ASSUME $x_{2.9}=1$, $y_{3.9}=1$, $y_{4.9}=1$ AND …)

QK	STEP	MEMORY	M				E				$N_{2,1}$		XA		X		XM	OPERATION
—	—	y	m_4	m_3	m_2	m_1	e_4	e_3	e_2	e_1	—	—	—	—	x_{j2}	x_{j1}		
01α	1										0	0						CLEAR $N_{2,1}$
02-11	2		y_4	y_3	y_2	y_1												READ
10α	3						0	0	0	0								CLEAR E
11β	4						0	0	0	0								CONFIGURATION (LOAD)
13α	5						y_4	y_3	0	0								
13β	6						y_3	0	0	y_4								
14α	7						y_3	0	0	y_4								SIGN EXTENSION
14β	8						y_3	1	1	y_4								
15α	9						y_3	1	1	y_4	1	y_4	$x_{j2}+1$	$x_{j1}+y_4$ *				LOAD $N_{2,1}$ WITH $\overline{E_{2,1}}$
16α	10						y_3	1	0	0								CLEAR $E_{2,1}$
18α	11						y_3	1	$x_{j2}+1$	$x_{j1}+y_4$								LOAD $E_{2,1}$ WITH $\overline{XA}$
18β	12						$x_{j1}+y_4$	y_3	1	$x_{j2}+1$								CONFIGURATION (STORE)
19α	13		$x_{j1}+y_4$	y_3	y_2	y_1												
21α	14						$x_{j1}+y_4$	y_3	y_2	y_1								ULTIMATE PULSE
21-31	15	$x_{j1}+y_4$ y_3 y_2 y_1																REWRITE

* THE + SIGNS ON THIS CHART INDICATE AN ARITHMETIC SUM AND NOT A LOGICAL SUM.

ADD INDEX (TO MEMORY)

PK	24	α		$\lfloor 00 \to PK$

	00	α	QI^{start} . . . . ⊃	$\lfloor start \diamond FK$
	01	α		$\lfloor 0 \to N_{2,1}$ $\lfloor 1 \to XAC$
	SEE QKM TIMING			
	09	α		
	10	α		$\lfloor 0 \to E$ $\lfloor 1 \to XAS$
	11	α		$\lfloor 13 \to QK$
		β		$\lfloor \bar{P} \to E$
	13	α		$M \xrightarrow[a,\,p]{0,1} E$
		β		$\lfloor P \to E$
QK	14	α		$\lfloor 0_{SE} \to E$ $\lfloor 1 \to XAC$
		β		$\lfloor C_{SE} \to E$
	15	α		$E_{2,1} \xrightarrow{1} N_{2,1}$
	16	α		$\lfloor 0 \to E_{2,1}$ $\lfloor 0 \to PI_1$
	17	α		
	18	α		$XA \xrightarrow{1} E_{2,1}$
		β		$\lfloor \bar{P} \to E$
	19	α	$\overline{MPA}$ . . . . ⊃	$E \xrightarrow[a,\,p]{0,1} M$
	20	α		
	21	α	$\overline{QKM^{VFF}}$ . . . . ⊃	$M \xrightarrow{0,1} E$ $\lfloor 0 \to EB$
	22	α		
	23	α		$\lfloor 0 \to EB$
	SEE QKM TIMING			
	31	α		

Op Class Decoder Lines Up: $PKIR^{def}$, $PKIR^{QK}$, $QKIR^{store}$

CAN 6-6-61

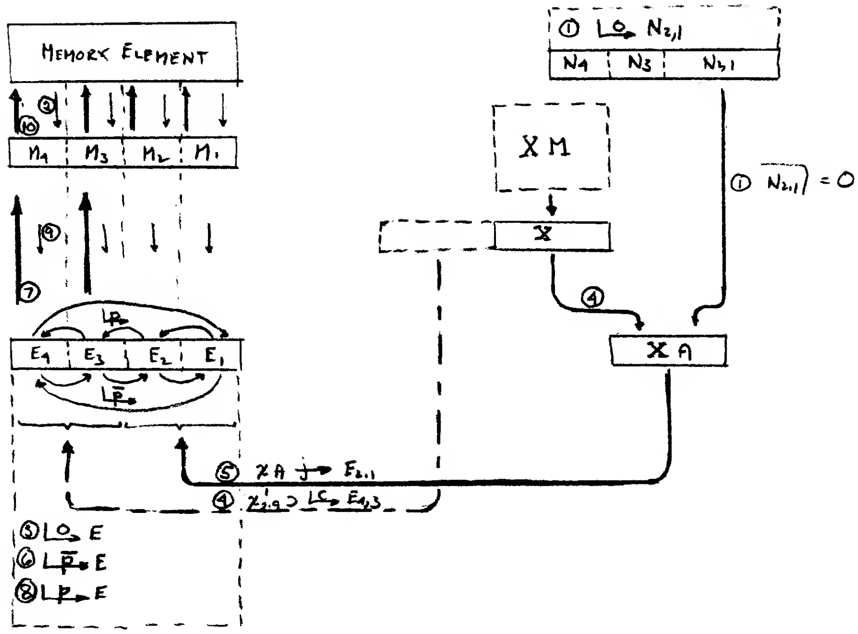
DEPOSIT INDEX (IN MEMORY)

OP CODE DESCRIPTION. DPX "deposits" the content of the specified index register (X_j) into the selected Memory Element register. DPX is a configurable, but non-indexable instruction.

SPECIAL FEATURES. The content of the X register is treated as an 18 bit signed word. By extending the sign of this word, a 36 bit word is formed. This 36 bit word then enters into the configuration process just as the 36 bit contents of A would enter into the configuration process during a STA.

DETAILS. After E is cleared, $E_{4,3}$ is complemented, based on the sign of the number in X. The content of X is then jammed into $E_{2,1}$. Note that the effect is as if X contained a 3rd and 4th quarter and the sign of X were extended into these quarters.

After E is loaded with the sign extended contents of XA, the normal configuration and storing process takes place as in a STE.



DPX ILLUSTRATIVE EXAMPLE (ASSUME $x_{2,9}=1$, $y_{2,9}=1$, $y_{4,9}=1$ AND)

QK	STEP	MEMORY	M				E				$N_{2,1}$		XA		X		XM	OPERATION
—	—	y	m_4	m_3	m_2	m_1	e_4	e_3	e_2	e_1	—	—	–	–	x_{j2}	x_{j2}		
01α	①										0	0						CLEAR $N_{2,1}$
02-11	②		y_4	y_3	y_2	y_1												READ
10α	③						0	0	0	0								CLEAR E
10β	④						1	1	0	0			x_{j2}	x_{j1}				EXTEND SIGN x_j
11α	⑤						1	1	x_{j2}	x_{j1}								LOAD $E_{2,1}$ WITH $\overline{XA}$
11β	⑥						x_{j1}			x_{j2}								CONFIGURATION
13α	⑦		x_{j1}	1	y_2	y_1												
13β	⑧						1	1	x_{j2}	x_{j1}								
21α	⑨						x_{j1}	1	y_2	y_1								ULTIMATE PULSE
21-31	⑩	x_{j1} 1 y_2 y_1																REWRITE

DEPOSIT INDEX (IN MEMORY)

PK	24	α		$\lfloor 00 \rightarrow$ PK

	00	α	QI^{start} . . . . ⊃	$\lfloor start \rightarrow$ FK
	01	α		$\lfloor 0 \rightarrow N_{2,1}$ $\lfloor 1 \rightarrow$ XAC
			SEE QKM TIMING	
	09	α		$\lfloor 0 \rightarrow PI_1$
	10	α		$\lfloor 0 \rightarrow$ E $\lfloor 1 \rightarrow$ XAS
		β	$X'_{2,9}$ . . . . ⊃	$\lfloor C \rightarrow E_{4,3}$
	11	α		$XA \xrightarrow{1} E_{2,1}$
		β		$\lfloor \bar{P} \rightarrow$ E
QK	12	α		
	13	α	$\overline{MPA}$ . . . . ⊃	$E \xrightarrow[\alpha,\beta]{0,1} M$
		β		$\lfloor P \rightarrow$ E
	14	α		$\lfloor 0 \rightarrow PI_1$ $\lfloor 21 \rightarrow$ QK
	21	α	$\overline{QKM^{VFF}}$ . . . . ⊃	$M \xrightarrow{0,1}$ $\lfloor 0 \rightarrow$ EB
	22	α		
	23	α		$\lfloor 0 \rightarrow$ EB
			SEE QKM TIMING	
	31	α		

OP Class Decoder Lines Up: $PKIR^{def}$, $PKIR^{QK}$, $QKIR^{X}$, $QKIR^{st}$, $QKIR^{store}$

CAN 6-6-61

OP CODE DESCRIPTION. SKM allows a programmer to select and use any bit in a memory word as an operand. This bit can be used to make a decision whether or not to SKIP. The bit can also be altered. Finally, the whole memory word can be rotated. SKM is a non-indexable and non-configurable instruction.

SPECIAL FEATURES. SKM has an extended PK cycle (PK^{25} through PK^{31}). PK waits in $PK^{25\alpha}$ until QK reaches $QK^{14\alpha}$. The J bits ($N_{3.6\ -\ 3.1}$) and CF bits ($N_{4.8\ -\ 4.4}$) are used for special purposes. The $E^{SKIP\ BIT}$ net samples the state of the selected bit. The operand can be rotated by an $E \xrightarrow[CYR]{1} M$ pulse.

DETAILS.

Operand Bit Selection. The N_J bits are used to select the operand bit. The scheme uses $N_{3.6}$ and $N_{3.5}$ to determine the quarter and $N_{3.4\ -\ 3.1}$ to select the bit in the quarter. In practice all the bit sampling and altering occurs in E_1; therefore, it is necessary for the operand to be read out, copied into E and the quarter containing the selected bit permuted into E_1 before the sampling occurs. The permutation is accomplished by transferring the content of $N_{3.6\ -\ 3.5}$ into $QKIR_{CF_{2-1}}$ and clearing $QKIR_{CF_{9-3}}$. CF_{9-4} then specifies a 36 bit fracture with all quarters active, while CF_{3-1} specifies the permutation required to place the selected quarter into E_1. The specific bit examined in E_1 is selected by $N_{3.4\ -\ 3.1}$.

In addition to examining the bits in E_1 certain other specific bits can be examined directly, e.g., $M_{4.10}$, MP and MP_{38}. In this case, the operand quarter specified by $N_{3.6\ -\ 3.5}$ has no logical significance.

Decision Logic. Three independent decisions are made based on the state of the $QKIR_{CF}$ bits.

CF_5 and CF_4	- determine the conditions for skipping.
CF_3	- determines whether or not the operand is to be rotated.
CF_2 and CF_1	- determine whether or not the selected bit is to be altered.

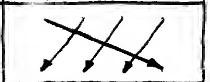
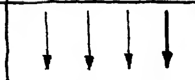
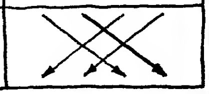
(See accompanying DECISION LOGIC tables.)

Note that the execution logic allows M_P and MP_{38} to be sampled (i.e., sensed), but not altered. All the other selected bits may be sampled and/or altered.

Note also that the selected bit is first sensed, and then altered. The whole word is not rotated until afterwards.

SKIP ON MEMORY

QUARTER SELECTION

$N_{3.6}$	$N_{3.5}$	$QKIRcf_2$	$QKIRcf_1$	QUARTER SELECTED	PERMUTATION DECODED
0	0	1	1	4	
0	1	0	0	1	
1	0	0	1	2	
1	1	1	0	3	

BIT SELECTED

$N_{3.4}$	$N_{3.3}$	$N_{3.2}$	$N_{3.1}$	BIT DECODED
0	0	0	1	$E_{i.1}$
0	0	1	0	$E_{i.2}$
0	0	1	1	$E_{i.3}$
0	1	0	0	$E_{i.4}$
0	1	0	1	$E_{i.5}$
0	1	1	0	$E_{i.6}$
0	1	1	1	$E_{i.7}$
1	0	0	0	$E_{i.8}$
1	0	0	1	$E_{i.9}$
1	0	1	0	$M_{4.10}$
1	0	1	1	M_P
1	1	0	0	MP_{38}

EXAMPLES OF COMPLETE BIT SELECTION:

$N^{110010}_{3.6-3.1} \supset E_{3.2}$

$N^{010011}_{3.6-3.1} \supset E_{1.3}$

$MP^{ev}_{38} =$ "0"
$MP^{odd}_{38} =$ "1"

OPERAND BIT SELECTION LOGIC

1

"SKIP" LOGIC @ $PK^{31\alpha}$

$PKIRcf_5$	$PKIRcf_4$	Pulse
0	0	——
0	1	$\overline{P}^7+1 \rightarrow P$
1	0	$E^0_{i.j} \supset \overline{P}^7+1 \rightarrow P$
1	1	$E^1_{i.j} \supset \overline{P}^7+1 \rightarrow P$

3

"ROTATE" LOGIC @ $QK^{19\alpha}$

$PKIRcf_3$	Pulse
0	$E \xrightarrow[a,P]{0,1} M$
1	$E \xrightarrow[cyr]{1} M$

$E_{i.j}$ = SELECTED BIT
$M_{4.10}, M_P, MP_{38}$ = SELECTED BIT
M_P & MP_{38} MAY BE SENSED BUT UNLIKE THE OTHER SELECTED BITS MAY NOT BE ALTERED.

2

"MAKE" LOGIC @ $QK^{18\alpha}$

$PKIRcf_2$	$PKIRcf_1$	Pulse
0	0	——
0	1	$\lfloor^{C} \rightarrow E_{i.j}$
1	0	$\lfloor^{0} \rightarrow E_{i.j}$
1	1	$\lfloor^{1} \rightarrow E_{i.j}$

DECISION EXAMPLES: (ASSUME $N^{110010}_{3.6-3.1}$, i.e. $E_{3.2}$ IS SELECTED BIT)

$$PKIRcf^{11111}_{5-1} \supset (E^1_{3.2} \supset \overline{P}^7+1 \rightarrow P;\ \lfloor^{1} \rightarrow E_{3.2};\ E \xrightarrow[cyr]{1} M)$$

$$PKIRcf^{01001}_{5-1} \supset (\ \overline{P}^7+1 \rightarrow P;\ \lfloor^{C} \rightarrow E_{3.2};\ E \xrightarrow[a,P]{0,1} M)$$

DECISION LOGIC

C.A.N. 6-7-61

SKIP ON MEMORY

			Condition		Action
PK	24	α			
	25	α	$QK^{14\alpha} \cdot QKIR^{skm}$	$\supset$	$\lfloor 31 \rightarrow PK$, $\overline{PK'}+1 \not\rightarrow PK$
	31	α	$PI^{ch\ seq}$	$\supset$	$\lfloor 1 \rightarrow PI_3$
			$E^{skip\ bit}$	$\supset$	$\overline{P'}+1 \rightarrow P$

			Condition		Action
	00	α			
	01	α			$\lfloor 0 \rightarrow N_{2,1}$
					$\lfloor 1 \rightarrow XAC$
			$PKIR^{skm}$	$\supset$	$N_{3.6,3.5} \xrightarrow{j} QKIR_{cf_{2,1}}$, $\lfloor 0 \rightarrow QKIR_{cf_{9-3}}$
			SEE QKM TIMING		
	09	α			
	10	α			
	11	α			$\lfloor 13 \rightarrow QK$
	13	α			$M \xrightarrow[a,p]{0,1} E$
		β			$\lfloor P \rightarrow E$
QK	14	α			$\lfloor 18 \rightarrow QK$, $\lfloor 31 \rightarrow PK$
					$\lfloor 0 \rightarrow PI_1$
	18	α	$PKIR_{cf_3^1} \cdot \overline{MPA}$	$\supset$	$\lfloor 0 \rightarrow M_{\overline{4\cdot 10}}$
			$PKIR_{cf_2^1} \cdot PKIR_{cf_1^0}$	$\supset$	$\lfloor 0 \rightarrow E_{1\cdot j}$
			$PKIR_{cf_2^0} \cdot PKIR_{cf_1^1}$	$\supset$	$\lfloor c \rightarrow E_{1\cdot j}$
			$PKIR_{cf_2^1} \cdot PKIR_{cf_1^1}$	$\supset$	$\lfloor 1 \rightarrow E_{1\cdot j}$
		β			$\lfloor \overline{P} \rightarrow E$
	19	α	$PKIR_{cf_3^0} \cdot \overline{MPA}$	$\supset$	$E \xrightarrow[a,p]{0,1} M$
			$PKIR_{cf_3^1} \cdot (MPAL_{sup} + MPAL^0)$	$\supset$	$E \xrightarrow[cyr]{1} M$
	20	α			
	21	α			$M \xrightarrow{0,1} E$
			$\overline{QKM^{VFF}}$	$\supset$	$\lfloor 0 \rightarrow EB$
	22	α			
	23	α			$\lfloor 0 \rightarrow EB$
			SEE QKM TIMING		
	31	α			

OP Decoder Lines Up: $PKIR^{def}$, $PKIR^{dis}$, $PKIR^{QK}$ & $QKIR^{store}$

$E^{skip\ bit} = [PKIR_{cf_4^0} \cdot (\overline{E_j} \cdot PKIR_{cf_5^1})] + [(PKIR_{cf_4^1} \cdot E_j) + (PKIR_{cf_4^1} \cdot PKIR_{cf_5^0})]$

E_j = selected bit is a "one". $\overline{E_j}$ = selected bit is a "zero".

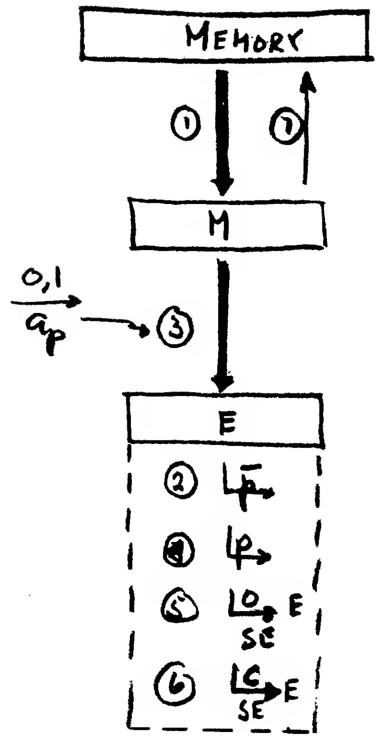
CAN 6-6-61

LOAD E (FROM MEMORY)

OP CODE DESCRIPTION. LDE "loads" the content of the selected Memory Element register into the E register. LDE is a configurable and indexable instruction.

SPECIAL FEATURES. The "ultimate pulse", which normally copies the content of the memory word into E in $QKIR^{LOAD}$ type instructions, does not occur.

DETAILS. See the description of the LDA (B, C and D) OP codes for an explanation of the basic "loading" process. The execution logic for LDE is similar to that for the other LD- OP codes, except that the transfers copying the content of the specified register into E and vice versa are omitted since E is the specified register, and the "ultimate pulse" does not occur.



LDE ILLUSTRATIVE EXAMPLE					
QK	STEP	MEMORY	M	E	OPERATION
—	—	y	m	e	
02 – 11	1	0	y	e	READ
11β	2	0	y	$e_{\bar{p}}$	CONFIGURATION
13α	3	0	y	$e_{\bar{p}} \mid y_{PR}$	
13β	4	0	y	$e \mid y_{CF}$	
14α	5	0	y	$e \mid y_{CF_{SE}}$	SIGN EXTENSION
14β	6	0	y		
21 – 31	7	y	y	$e \mid y_{CF_{SE}}$	REWRITE

LOAD E (FROM MEMORY)

PK	24	α	$\lfloor 00 \rightarrow$ PK

	00	α	QI^{start} . . . . $\supset$ $\lfloor start \rightarrow$ FK, $\lfloor 0 \rightarrow$ PI_1
	01	α	
	SEE QKM TIMING		
	09	α	
	10	α	
	11	α	$\lfloor 13 \rightarrow$ QK
		β	$\lfloor \bar{p} \rightarrow$ E
QK	13	α	$M \xrightarrow[\alpha,\beta]{0,1} E$
		β	$\lfloor p \rightarrow$ E
	14	α	$\lfloor 0 \xrightarrow[SE]{} E$ $\lfloor 21 \rightarrow$ QK
		β	$\lfloor c \xrightarrow[SE]{} E$
	21	α	
	22	α	
	23	α	$\lfloor 0 \rightarrow$ EB $\lfloor 31 \rightarrow$ QK
	31	α	

OP Decoder Lines Up: $PKIR^{def}$
$PKIR^{ind}$
$PKIR^{QK}$
$QKIR^{ld}$
$QKIR^{E}$
$QKIR^{load}$

C.A.N 6-6-61

SPECIFY CONFIGURATION (FROM MEMORY)

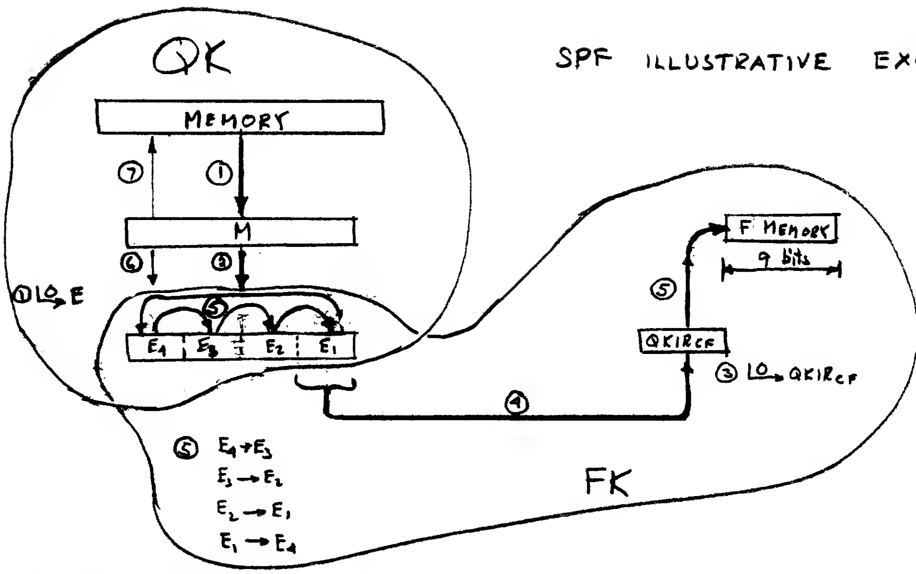
OP CODE DESCRIPTION. SPF "specifies" configuration by loading the content of quarter 1 of the selected Memory Element register into the specified F Memory register. SPF is an indexable, but non-configurable instruction.

SPECIAL FEATURES. The FK counter controls E register pulses during part of the instruction.

DETAILS. The 36 bit operand word in the selected Memory Element register is placed in E. The FK cycle initiated in $QK^{13\alpha}$ then places the content of quarter 1 of E into the F Memory register specified by the $PKIR_{CF}$ bits.

The effect of the permuting in E that occurs during the FK cycle is nullified by the "ultimate pulse" that copies the content of M into E.

SPF ILLUSTRATIVE EXAMPLE



QK	FK	STEP	MEMORY	M	E (E_4 E_3 E_2 E_1)	$QKIR_{CF}$	F MEM	OPERATION
—		—	y*	m	e			
02-11		1		y	0			READ
13d		2		y	y			E SET UP

QK^{13d} ⊃ LSTART → FK

FK	STEP	MEMORY	M	E	$QKIR_{CF}$	F MEM	OPERATION
—	[illegible]			y_4 y_3 y_2 y_1		f	
0r-24	3				0		"LOAD" F WITH
	4				y_1		CONTENTS OF
	5			y_1 y_4 y_3 y_2	y_1	y_1	E_1, i.e. y_1

QK	FK	STEP	MEMORY	M	E	$QKIR_{CF}$	F MEM	OPERATION
—		—		y	~~y_4 y_3 y_2 y_1~~			
23		6		y	y			ULTIMATE PULSE
21-31		7	y	y	y			REWRITE

* $\boxed{y}$ = $\boxed{y_4\ y_3\ y_2\ y_1}$

SPECIFY CONFIGURATION (FROM MEMORY)

PK	24	α	$\lfloor 100 \rightarrow PK$

QK	00	α	
	01	α	
	SEE QKM TIMING		
	09	α	
	10	α	$\lfloor 0 \rightarrow E$
	11	α	$\lfloor 13 \rightarrow QK$
	13	α	$\lfloor start \diamond FK$ $M \xrightarrow{0,1} E$
	14	α	$\lfloor 21 \rightarrow QK$
	21	α	$\lfloor 0 \rightarrow PI_1$
	22	α	
	23	α	$M \xrightarrow{0,1} E$ $\lfloor 0 \rightarrow EB$ $\lfloor 31 \rightarrow QK$
	31	α	

OP Class Decoder Lines Up: $PKIR^{def}$
$PKIR^{QK}$
$PKIR^{f}$
$PKIR^{lf}$
$QKIR^{load}$
$QKIR^{specify}$

C.A.N 6-6-61

SPECIFY GROUP (OF FOUR CONFIGURATIONS FROM MEMORY)

OP CODE DESCRIPTION. SPG "specifies" a group of four configurations by loading the content of the selected Memory Element register into four successive F Memory registers. SPG is an indexable, but non-configurable instruction.

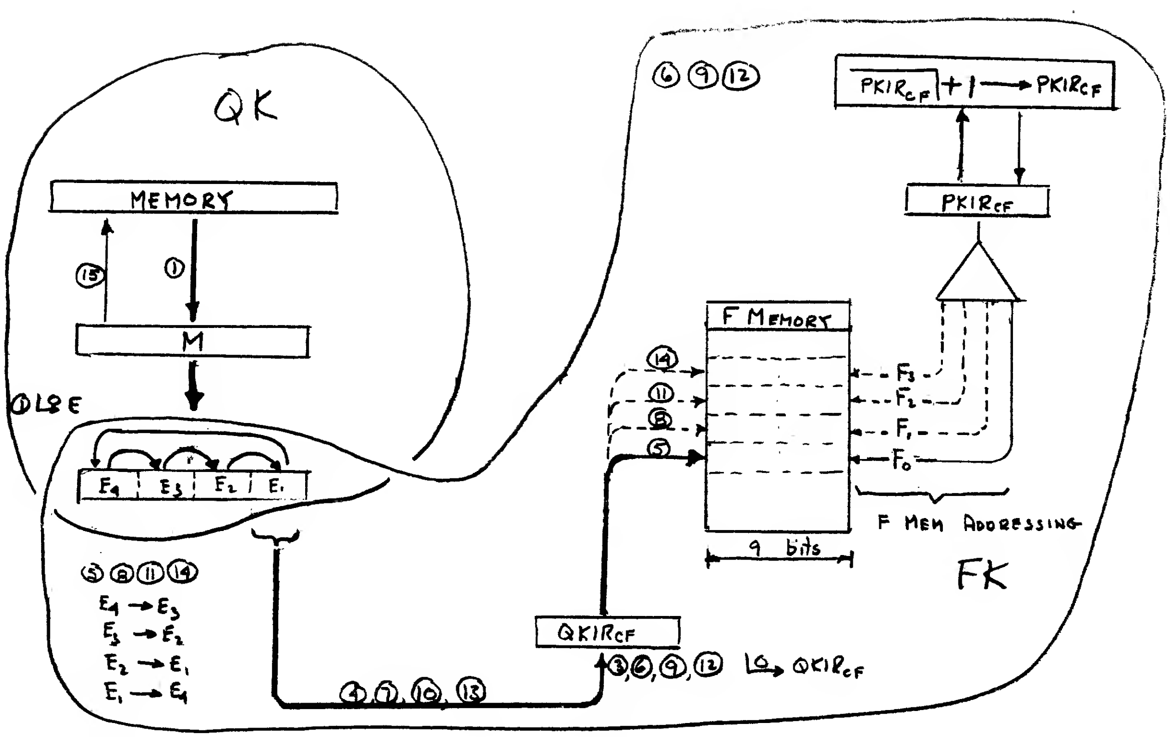
SPECIAL FEATURES. $PKIR_{CF}$ specifies the initial address of four successive registers in the F Memory. $PKIR_{CF}$ is indexed three times. Quarter-wise shifting to the right occurs in E during the instruction. FK controls E register pulses during part of the instruction.

DETAILS. The 36 bit operand word in the selected Memory Element register is placed in E. The FK cycle initiated in $QK^{13\alpha}$ repeats four times the process of storing the content of E_1 into the specified F Memory register. y_1 (see attached figure) is stored in the F Memory register specified by the CF bits originally transferred from $N_{4.8 - 4.1}$ to $PKIR_{CF}$. (In the figure, these bits select register F_0 in the F Memory.)

Before the first FK iteration, $PKIR_{CF}$ is inhibited from indexing. However, before the second FK iteration, $PKIR_{CF}$ is indexed by one so that it selects the next F Memory register.

After the transfer between E_1 and the F Memory, the content of E is shifted quarter wise to the right. Thus, in the second iteration y_2 is stored in the F Memory.

At the end of four iterations, E contains the original operand word so that no "ultimate pulse" need occur.



QK	FK	STEP	MEMORY	M	E (E_4 E_3 E_2 E_1)	QKIR$_{CF}$	F MEMORY (F_3 F_2 F_1 F_0)	PKIR$_{CF}$ (F MEM ADD)	OPERATION
—		—	y*	m	e				
02-11		1		y	0				READ
13d		2		y	y				E SET UP

QK^{13d} ⊃ START FK

QK	FK	STEP	MEMORY	M	E	QKIR$_{CF}$	F MEMORY	PKIR$_{CF}$	OPERATION
	—	—			y_4 y_3 y_2 y_1		f_0 f_1 f_2 f_3	F_0	
	0-1	3				0		F_0	"LOAD" F_0 with
		4				y_1		F_0	CONTENTS OF
		5			y_1 y_4 y_3 y_2	y_1	y_1 f_1 f_2 f_3	F_0	E_1, i.e. y_1
	2-3	6				0		$F_0+1=F_1$	"LOAD" F_1 WITH
		7				y_2		F_1	CONTENTS OF
		8			y_2 y_1 y_4 y_3	y_2	y_1 y_2 f_2 f_3	F_1	E_1, i.e. y_2
	4-5	9				0		$F_1+1=F_2$	"LOAD" F_2 WITH
		10				y_3		F_2	CONTENTS OF
		11			y_3 y_2 y_1 y_4	y_3	y_1 y_2 y_3 f_3	F_2	E_1, i.e. y_3
	6-7	12				0		$F_2+F_1=F_3$	"LOAD" F_3 WITH
		13				y_4		F_3	CONTENTS OF
		14			y_4 y_3 y_2 y_1	y_4	y_1 y_2 y_3 y_4	F_3	E_1, i.e. y_4

QK	FK	STEP	MEMORY	M	E	QKIR$_{CF}$	F MEMORY	PKIR$_{CF}$	OPERATION
—		—		y	y	y_4			
21-31		15	y	y	y	y_4			REWRITE

* $\boxed{y}$ = $\boxed{y_4\ y_3\ y_2\ y_1}$

SPECIFY GROUP (OF FOUR CONFIGURATIONS FROM MEMORY)

PK	24	α	$\underline{\mid 00} \rightarrow$ PK

	00	α	
	01	α	
	SEE QKM TIMING		
	09	α	
	10	α	$\underline{\mid 0} \rightarrow$ E
	11	α	$\underline{\mid 13} \rightarrow$ QK
QK	13	α	$\underline{\mid \text{start}} \diamond$ FK $M \xrightarrow{0,1} E$
	14	α	$\underline{\mid 21} \rightarrow$ QK
	21	α	$\underline{\mid 0} \rightarrow PI_1$
	22	α	
	23	α	$\underline{\mid 31} \rightarrow$ QK
	31	α	

OP Class Decoder Lines Up:

$PKIR^{def}$
$PKIR^{QK}$
$PKIR^{f}$
$PKIR^{lf}$
$PKIR^{ff}$
$QKIR^{load}$
$QKIR^{specify}$

C.A.N 6-6-61

LOAD A, B, C, D (FROM MEMORY)

OP CODE DESCRIPTION. LD- "loads" the specified Arithmetic Element register with the content of the selected Memory Element register. These are configurable and indexable instructions.

SPECIAL COMMENT. The execution logic for these instructions is found, in modified form, in all the $\mathrm{QKIR}^{\mathrm{LOAD}}$ type instructions.

DETAILS. The basic "loading" process consists of:

1) "Reading" the content of the selected Memory Element register into M. Slight variations will occur in this process depending on which memory register is selected.

2) Loading E with the content of the specified Arithmetic Element register. This is necessary in order that the configuration operation which follows will not disturb the inactive quarters of the specified Arithmetic Element register.

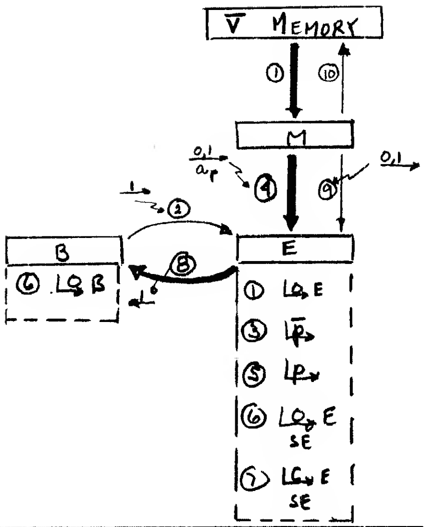
3) Configuring the operand. This consists of: (1) inversely permuting E, (2) transferring the content of M into E under "permuted activity" control, and (3) directly permuting the content of E.

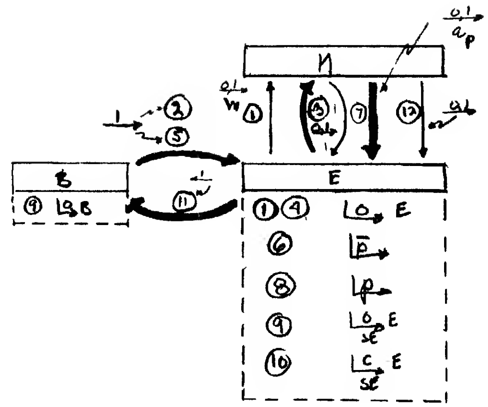
4) Extending the sign of the configured operand. This is accomplished by a clear and complement operation under "sign extension" control.

5) Loading the specified Arithmetic Element register with the content of E.

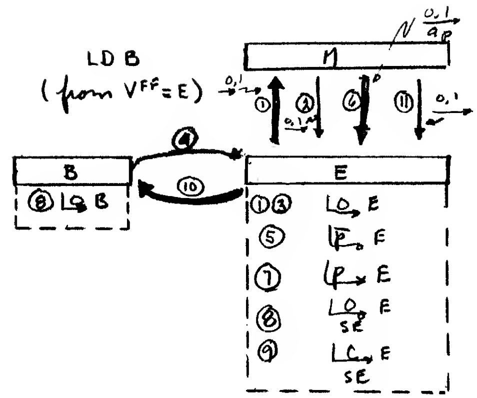
6) Firing off an "ultimate pulse". This copies the original operand word from M into E.

7) Rewriting the original operand back into the selected Memory Element register. In the case of the V Memory, a rewrite phase is not necessary since the readout is not destructive.





LDB (from VFF = E)



LDB From $\overline{V}$ ILLUSTRATIVE EXAMPLE						
QK	STEP	$\overline{V}$ MEMORY	M	E	B	OPERATION
—	—	y	m	e	b	
02-11	1	y	y	0	b	READ
11α	2	0	y	b	b	E SetUp
11β	3	0	y	$b_{\bar{p}}$	b	CONFIGURATION
13α	4	0	y	$b_{\bar{p}} \mid y_{PA}$	b	
13β	5	0	y	$b \mid y_{CF}$	b	
14α	6	0	y	$b \mid y_{CF_{SE}}$	0	SIGN EXTENSION
14β	7	0	y		0	
21α	8	0	y	$b \mid y_{CF_{SE}}$	$b \mid y_{CF_{SE}}$	"Load B"
23α	9	0	y	y	$b \mid y_{CF_{SE}}$	ULTIMATE PULSE
21-31	10	y	y	y	$b \mid y_{CF_{SE}}$	REWRITE

LDB FROM B ILLUSTRATIVE EXAMPLE					
QK	STEP	B	M	E	OPERATION
—	—	b	m	e	
2α	1	b	e	0	M SET UP
3α	2	b	e	b	
9α	3	b	b	e	
10α	4	b	b	0	
11α	5	b	b	b	E SET UP
11β	6	b	b	$b_{\bar{p}}$	CONFIGURATION
13α	7	b	b	$b_{\bar{p}} \mid b_{PA}$	
13β	8	b	b	$b \mid b_{CF}$	
14α	9	0	b	$b \mid b_{CF_{SE}}$	SIGN EXTENSION
14β	10	0	b		
21α	11	$b \mid b_{CF_{SE}}$	b	$b \mid b_{CF_{SE}}$	"Load B"
23α	12	$b \mid b_{CF_{SE}}$	b	b	ULTIMATE PULSE

LDB FROM E ILLUSTRATIVE EXAMPLE					
QK	STEP	B	M	E	OPERATION
—	—	b	m	e	
2α	1	b	e	0	M SET UP
3α	3	b	e	e	
10α	3	b	e	0	
11α	4	b	e	b	E SET UP
11β	5	b	e	$b_{\bar{p}}$	CONFIGURATION
13α	6	b	e	$b_{\bar{p}} \mid e_{PA}$	
13β	7	b	e	$b \mid e_{CF}$	
14α	8	0	e	$b \mid e_{CF_{SE}}$	SIGN EXTENSION
14β	9	0	e		
21α	10	$b \mid e_{CF_{SE}}$	e	$b \mid e_{CF_{SE}}$	"Load B"
23α	11	$b \mid e_{CF_{SE}}$	e	e	ULTIMATE PULSE

LDA(B, C and D) (24 (5, 6 and 7))

			LOAD A, B, C, D (FROM MEMORY)	24 LDA	25 LDB	26 LDC	27 LDD

PK	24	α		$\lfloor 100 \to PK$

QK	00	α	QI^{start} . . . . $\supset$	$\lfloor start \to FK$, $\lfloor 0 \to PI_1$
	01	α		
			SEE QKM TIMING	
	09	α		
	10	α		$\lfloor 0 \to E$
	11	α		$\lfloor 13 \to QK$
			$QKIR^{lda}$ . . . . $\supset$	$A \xrightarrow{1} E$
			$QKIR^{ldb}$ . . . . $\supset$	$B \xrightarrow{1} E$
			$QKIR^{ldc}$ . . . . $\supset$	$C \xrightarrow{1} E$
			$QKIR^{ldd}$ . . . . $\supset$	$D \xrightarrow{1} E$
		β		$\lfloor \bar{P} \to E$
	13	α		$M \xrightarrow[\alpha,\beta]{0,1} E$
		β		$\lfloor P \to E$
	14	α		$\lfloor 21 \to QK$
				$\lfloor 0 \xrightarrow[SE]{} E$
			$QKIR^{lda}$ . . . . $\supset$	$\lfloor 0 \to A$
			$QKIR^{ldb}$ . . . . $\supset$	$\lfloor 0 \to B$
			$QKIR^{ldc}$ . . . . $\supset$	$\lfloor 0 \to C$
			$QKIR^{ldd}$ . . . . $\supset$	$\lfloor 0 \to D$
		β		$\lfloor C \xrightarrow[SE]{} E$
	21	α	$QKIR^{lda}$ . . . . $\supset$	$E \xrightarrow{1} A$
			$QKIR^{ldb}$ . . . . $\supset$	$E \xrightarrow{1} B$
			$QKIR^{ldc}$ . . . . $\supset$	$E \xrightarrow{1} C$
			$QKIR^{ldd}$ . . . . $\supset$	$E \xrightarrow{1} D$
	22	α		
	23	α		$M \xrightarrow{0,1} E$
				$\lfloor 0 \to EB$
				$\lfloor 31 \to QK$
	31	α		

OP Class Decoder Lines Up: $PKIR^{def}$, $PKIR^{ind}$, $PKIR^{QK}$, $QKIR^{ld}$, $QKIR^{load}$ & $PKIR^{AE}$

$QKIR^{lda} \supset QKIR^{A}$; $QKIR^{ldb} \supset QKIR^{B}$; $QKIR^{ldc} \supset QKIR^{C}$; $QKIR^{ldd} \supset QKIR^{D}$

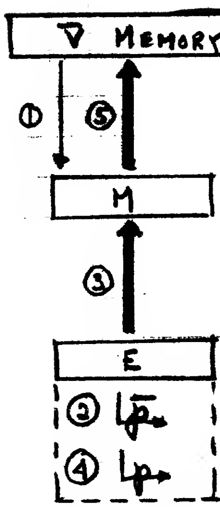
C.A.N 6-6-61

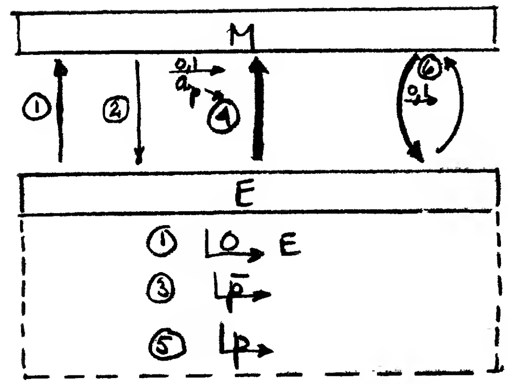
STORE E (IN MEMORY)

OP CODE DESCRIPTION. STE "stores" the content of the E register in the selected Memory Element register. STE is a configurable and indexable instruction.

SPECIAL FEATURES. The "ultimate pulse", which normally copies the word to be "stored" in memory also into the E register, does not occur.

DETAILS. See the description of the STA (B, C and D) OP codes for an explanation of the basic "storing" process. The execution logic for STE is similar to that for the other ST- OP codes, except that the transfers copying the content of the specified register into E and vice versa are omitted, since E is the specified register, and the "ultimate pulse" does not occur.





STE in $\overline{V}$ ILLUSTRATIVE EXAMPLE

QK	STEP	$\overline{V}$ MEMORY	M	E	OPERATION
—	—	y	m	e	
02-11	1	0	y	e	READ
11β	2	0	y	$e_{\bar{p}}$	CONFIGURATION
13α	3	0	$y \mid e_{CF}$	$e_{\bar{p}}$	
13β	4	0	$y \mid e_{CF}$	e	RESTORE E
21-31	5	$y \mid e_{CF}$	$y \mid e_{CF}$	e	REWRITE ("Store E")

STE in E ILLUSTRATIVE EXAMPLE

QK	STEP	E	M	OPERATION
—	—	e	m	
02α	1	0	e	M SET UP
09α	2	e	e	
11β	3	$e_{\bar{p}}$	e	CONFIGURATION
13α	4	$e_{\bar{p}}$	$e \mid e_{CF}$	
13β	5	e	$e \mid e_{CF}$	RESTORE E
21α	6	$e \mid e_{CF}$	e	"Store E" in E

STORE E (IN MEMORY)

PK	24	α		$\lfloor 00 \rightarrow PK$
	00	α	QI^{start} . . . . $\supset$	$\lfloor start \diamond FK$, $\lfloor 0 \rightarrow PI$,
	01	α		
			SEE QKM TIMING	
	09	α		
	10	α		
	11	β		$\lfloor \bar{P} \rightarrow E$
	12	α		
	13	α	$\overline{MPA}$ . . . . $\supset$	$E \xrightarrow[a,p]{0,1} M$
	13	β		$\lfloor P \rightarrow E$
QK	14	α		$\lfloor 21 \rightarrow QK$
	21	α	$\overline{QKM^{VFF}}$ . . . . $\supset$ QKM^{VFF} . . . . $\supset$ $QKM^{VFF} \cdot \overline{MPA}$ . . . . . . $\supset$	$\lfloor 0 \rightarrow EB$ $M \xrightarrow{0,1} E$ $E \xrightarrow{0,1} M$
	22	α		
	23	α	$QKM^{VFF} \cdot \overline{VMD^{E}}$ . . . . . . $\supset$	$\lfloor 0 \rightarrow EB$ $M \xrightarrow{0,1} E$
			SEE QKM TIMING	
	31	α		

OP Class Decoder Lines Up:

$PKIR^{def}$
$PKIR^{ind}$
$PKIR^{QK}$
$QKIR^{st}$
$QKIR^{store}$
$QKIR^{E}$

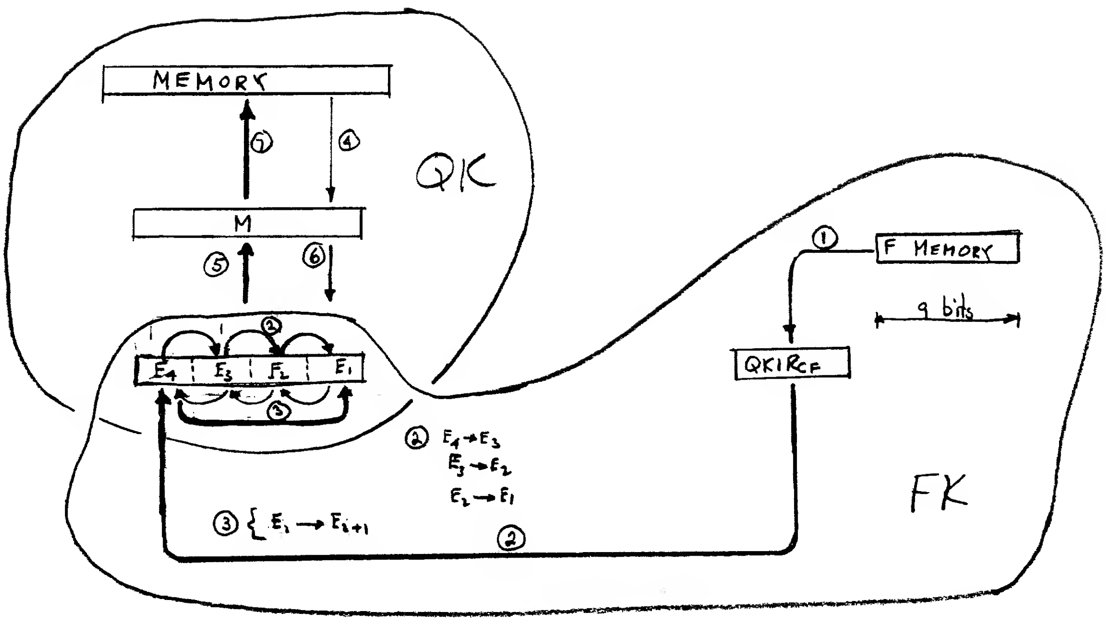
C.A.N 6-8-61

FILE CONFIGURATION (IN MEMORY)

OP CODE DESCRIPTION. FLF "files" the content of the specified F Memory register in the selected Memory Element register. FLF is an indexable, but non-configurable instruction.

SPECIAL FEATURES. An FK cycle is initiated during the PK cycle. The PK counter controls the E register pulses during part of the instruction. Quarter wise shifting to the right occurs in E.

DETAILS. The FK cycle, initiated by $PK^{13\alpha}$, shifts the content of E quarter wise to the right. The content of the F Memory register selected by the $PKIR_{CF}$ bits is then loaded into E_4. The content of E is then shifted quarter wise to the left. This leaves the content of the selected F Memory register in E_1. The FK counter then stops and the QK counter starts. At $QK^{13\alpha}$ the content of E_1 is transferred into M_1. This places the content of the selected F Memory register in M_1 and leaves M_{4-2} with its original content. The content of M is then written in memory.



$PK^{13d} \supset \underrightarrow{\lfloor 0}$ FI; $FI^0 \cdot EB^0 \cdot PKIR^{SF} \supset \underrightarrow{\text{START}}$ FK

FK	STEP	MEMORY	M	E: E_4	E_3	E_2	E_1	$QKIR_{CF}$	F MEMORY	OPERATION
–	–			e_4	e_3	e_2	e_1		f	
0-2	①							f		LOAD E_1 WITH
	②			f	e_4	e_3	e_2	f	f	CONTENTS OF
	③			e_4	e_3	e_2	f	f	f	F, i.e f

$FK^{2d} \supset \underrightarrow{\lfloor 1}$ FI; $FI^1 \cdot PI_1^1 \cdot QKS^0 \cdot START_2^1 = QI^{START}$

QK						
–	–	y	m	e_4 e_3 e_2 f		
02-11	④		y_4 y_3 y_2 y_1	e_4 e_3 e_2 f		READ
13d	⑤		f	e_4 e_3 e_2 f		M. LOADED WITH f
21d	⑥		f	y_4 y_3 y_2 f		ULTIMATE PULSE
21-31	⑦	y_4 y_3 y_2 f	y_4 y_3 y_2 f	y_4 y_3 y_2 f		REWRITE

$\boxed{\quad y}$ = $\boxed{y_4 \,|\, y_3 \,|\, y_2 \,|\, y_1}$

FILE CONFIGURATION (IN MEMORY)

PK	24	α		$\xrightarrow{00}$ PK

QK	00	α	QI^{start} . . . . ⊃	$\xrightarrow{0}$ PI_1
	01	α		
	SEE QKM TIMING			
	09	α		
	10	α		
	11	α		
	12	α		
	13	α	$\overline{MPA}$ . . . . ⊃	$E_1 \xrightarrow{0,1} M_1$
	14	α		$\xrightarrow{21}$ QK
	21	α	$\overline{QKM^{VFF}}$ . . . . ⊃	$M \xrightarrow{0,1} E$ $\xrightarrow{0}$ EB
	22	α		
	23	α		$\xrightarrow{0}$ EB
	SEE QKM TIMING			
	31	α		

OP Class Decoder Lines Up:

$PKIR^{def}$
$PKIR^{QK}$
$PKIR^{f}$
$PKIR^{sf}$
$QKIR^{store}$
$QKIR^{file}$

C.A.N 6-8-61

FILE GROUP (OF FOUR CONFIGURATIONS IN MEMORY)

OP CODE DESCRIPTIONS. FLG "files" a group of four successive F Memory words in a single register in the Memory Element. FLG is an indexable, but non-configurable instruction.

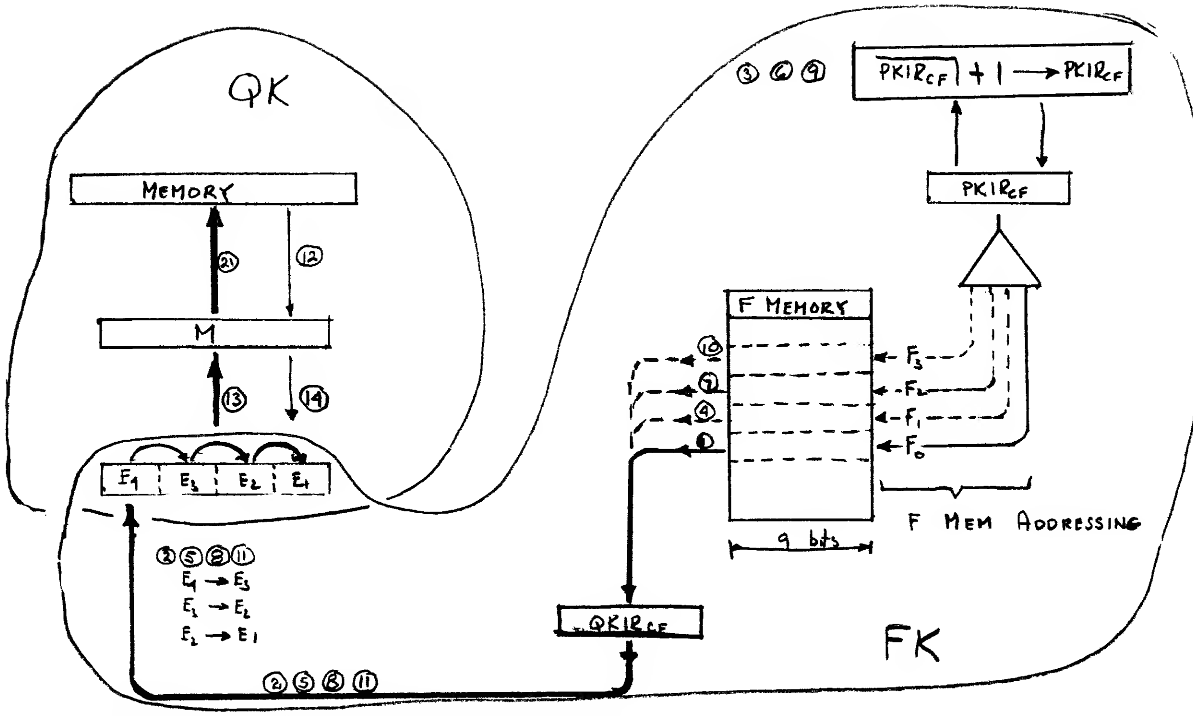
SPECIAL FEATURES. An FK cycle is initiated during the PK cycle. The FK counter controls E register pulses during part of the instruction. $PKIR_{CF}$ is indexed three times. Quarter-wise shifting to the right occurs in E.

DETAILS. The FK cycle, initiated by $PK^{13\alpha}$, repeats four times the basic process of loading E with the content of an F Memory register. The first word read out of the F Memory comes from the register selected by the CF bits. The content of E is shifted quarter wise to the right before the content of $QKIR_{CF}$ is copied into E_4. (The content of E_1 is not shifted and is lost.)

Before the first FK iteration, $PKIR_{CF}$ is inhibited from indexing. However, before the second FK iteration, $PKIR_{CF}$ is indexed by one so that it selects the next F Memory register. The content of E is again shifted quarter wise to the right and the new content of $QKIR_{CF}$ is then loaded into E_4.

At the end of four iterations, E contains the contents of four successive F Memory registers with the first in E_1, the second in E_2, f_3, etc.

After the FK counter has loaded E with the content of four registers in the F Memory, FK stops running. The QK counter then starts and stores the contents of E in the selected Memory Element register.



$PK^{13d} \supset L_0 \rightarrow FI\ ;\ FI^{\circ} \cdot EB^{\circ} \cdot PKIR^{SF} \supset \underline{START}\ FK$

FK	STEP	MEMORY	M	E (E_4 E_3 E_2 E_1)	$QKIR_{CF}$	F MEMORY (F_0 F_1 F_2 F_3)	$PKIR_{CF}$ (F MEM ADD)	OPERATION
—	—			e_4 e_3 e_2 e_1		f_0 f_1 f_2 f_3	F_0	
0-1	1				f_0		F_0	LOAD E_4 WITH CONTENTS OF F_0, i.e f_0
	2			f_0 e_4 e_3 e_1	f_0	f_0	F_0	
2-3	3				f_0		$F_0 + 1 = F_1$	LOAD E_4 WITH CONTENTS OF F_1, i.e f_1
	4				f_1	f_1	F_1	
	5			f_1 f_0 e_4 e_3	f_1		F_1	
4-5	6				f_1		$F_1 + 1 = F_2$	LOAD E_4 WITH CONTENTS OF F_2, i.e f_2
	7				f_2	f_2	F_2	
	8			f_2 f_1 f_0 e_4	f_2		F_2	
6-7	9				f_2		$F_2 + 1 = F_3$	LOAD E_4 WITH CONTENTS OF F_3, i.e f_3
	10				f_3	f_3	F_3	
	11			f_3 f_2 f_1 f_0	f_3	f_0 f_1 f_2 f_3	F_3	

$FK^{7d} \supset LL \rightarrow FI\ ;\ FI' \cdot PI'_1 \cdot QKS^{\circ} \cdot START'_2 = QJ^{START}$

QK						
—	—	y	m	f*		
02-11	12		y			READ
13d	13		f			M SET UP
21d	14		f			ULTIMATE PULSE
21-31	15	f	f	f		REWRITE

$\boxed{f} = \boxed{f_3 \,\vdots\, f_2 \,\vdots\, f_1 \,\vdots\, f_0}$

FILE GROUP (OF FOUR CONFIGURATIONS IN MEMORY)

PK	24	α		$\underline{100} \rightarrow$ PK
QK	00	α	QI^{start} . . . . ⊃	$\underline{0} \rightarrow PI_1$
	01	α		
	SEE QKM TIMING			
	09	α		
	10	α		
	11	α		
	12	α		
	13	α	$\overline{MPA}$ . . . . ⊃	$E \xrightarrow{0,1} M$
	14	α		$\underline{21} \rightarrow$ QK
	21	α	$\overline{QKMFF}$ . . . . ⊃	$M \xrightarrow{0,1} E$ $\underline{0} \rightarrow$ EB
	22	α		
	23	α		$\underline{0} \rightarrow$ EB
	SEE QKM TIMING			
	31	α		

OP Class Decoder Lines Up:

$PKIR^{def}$
$PKIR^{QK}$
$PKIR^{f}$
$PKIR^{sf}$
$PKIR^{ff}$
$QKIR^{store}$
$QKIR^{file}$

STORE A, B, C, D (IN MEMORY)

OP CODE DESCRIPTION. ST- "stores" the content of the specified Arithmetic Element register in the selected Memory Element register. These are indexable and configurable instructions.

SPECIAL COMMENT. The basic execution logic of these instructions is found in modified form in all the $\text{QKIR}^{\text{STORE}}$ type instructions.

DETAILS. The basic "storing" process consists of:

1) "Reading" the content of the selected Memory Element register into M. Slight variations will occur in this process depending on which memory register is selected.

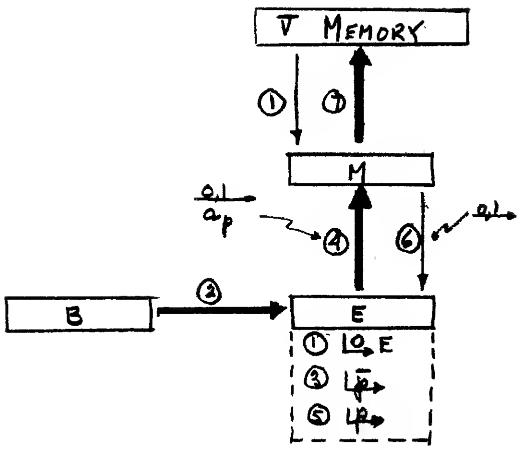
2) Loading E with the content of the specified Arithmetic Element register.

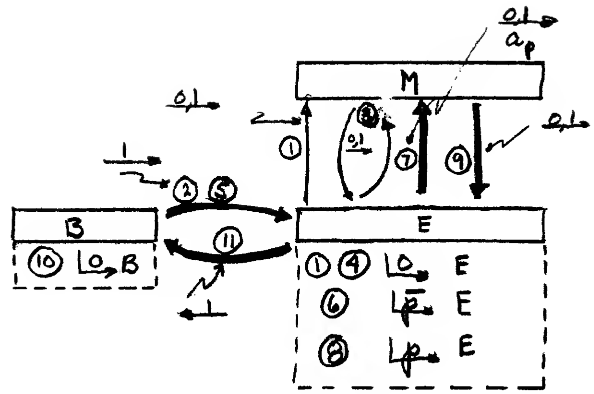
3) Configuring the content of the specified Arithmetic Element register. This consists of inversely permuting the content of E and then transferring the content of E into M under permuted activity control.

4) Restoring the content of E. This is done by a direct permutation pulse. In STA (B, C, D) this is an unnecessary step, since the effect is wiped out by the succeeding "ultimate" pulse, but it is used by certain OP codes (e.g., STE) which make use of the basic store process.

5) Firing off an "ultimate pulse". This copies the word being stored in memory into E.

6) Writing, i.e., "storing", the content of M in the selected Memory Element register. This process will vary depending on the memory selected.





STB in $\overline{V}$ ILLUSTRATIVE EXAMPLE						
QK	STEP	$\overline{V}$ MEMORY	M	E	B	OPERATION
—	—	y	m	e	b	
02-11	1	0	y	0	b	READ
11α	2	0	y	b	b	E SET UP
11β	3	0	y	$b_{\bar{p}}$	b	CONFIGURATION
13α	4	0	$y \mid b_{CF}$	$b_{\bar{p}}$	b	
13β	5	0	$y \mid b_{CF}$	b	b	RESTORE E
21α	6	0	$y \mid b_{CF}$	$y \mid b_{CF}$	b	ULTIMATE PULSE
21-31	7	$y \mid b_{CF}$	$y \mid b_{CF}$	$y \mid b_{CF}$	b	REWRITE ("Store B")

STB in B ILLUSTRATIVE EXAMPLE					
QK	STEP	B	M	E	OPERATION
—	—	b	m	e	
2α	1	b	e	0	M SET UP
3α	2	b	e	b	
9α	3	b	b	e	
10α	4	b	b	0	
11α	5	b	b	b	E SET UP
11β	6	b	b	$b_{\bar{p}}$	CONFIGURATION
13α	7	b	$b \mid b_{CF}$	$b_{\bar{p}}$	
13β	8	b	$b \mid b_{CF}$	b	RESTORE E
21α	9	b	$b \mid b_{CF}$	$b \mid b_{CF}$	ULTIMATE PULSE
22α	10	0	$b \mid b_{CF}$	$b \mid b_{CF}$	"Store B" in B *
23α	11	$b \mid b_{CF}$	$b \mid b_{CF}$	$b \mid b_{CF}$	

* STB in E is the same as STB in B except STEPS 10 and 11 are omitted, i.e. the register manipulations are completed with the ULTIMATE PULSE

STA (B, C and D) (34 (5, 6 and 7))

STORE A, B, C, D (IN MEMORY)

	34	35	36	37
	STA	STB	STC	STD

PK

24	α		$\lfloor 00 \rightarrow$ PK

QK

00	α	QI^{start} . . . . ⊃	$\lfloor start \rightarrow$ FK, $\lfloor 0 \rightarrow$ PI,
01	α		
SEE QKM TIMING			
09	α		
10	α		$\lfloor 0 \rightarrow$ E
11	α	$QKIR^{sta}$ . . . . ⊃ $QKIR^{stb}$ . . . . ⊃ $QKIR^{stc}$ . . . . ⊃ $QKIR^{std}$ . . . . ⊃	$A \xrightarrow{1} E$ $B \xrightarrow{1} E$ $C \xrightarrow{1} E$ $D \xrightarrow{1} E$
	β		$\lfloor \overline{P} \rightarrow$ E
12	α		
13	α	$\overline{MPA}$ . . . . ⊃	$E \xrightarrow[a,p]{0,1} M$
	β		$\lfloor P \rightarrow$ E
14	α		$\lfloor 21 \rightarrow$ QK
21	α	$\overline{QKM^{VFF}}$ . . . . ⊃	$M \xrightarrow{0,1} E$ $\lfloor 0 \rightarrow$ EB
22	α		
23	α		$\lfloor 0 \rightarrow$ EB
SEE QKM TIMING			
31	α		

OP Class Decoder Lines Up:

$PKIR^{def}$
$PKIR^{ind}$
$PKIR^{QK}$
$QKIR^{store}$
$QKIR^{st}$
$PKIR^{AE}$

$QKIR^{sta}$ ⊃ $QKIR^{A}$
$QKIR^{stb}$ ⊃ $QKIR^{B}$
$QKIR^{stc}$ ⊃ $QKIR^{C}$
$QKIR^{std}$ ⊃ $QKIR^{D}$

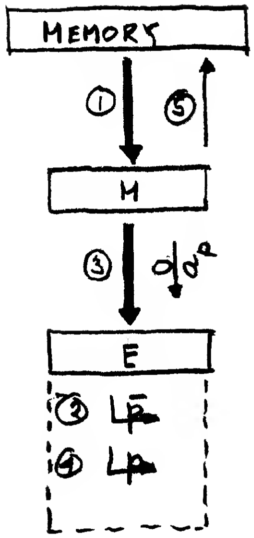
C.A.N 6-8-61

INTERSECT E (WITH MEMORY)

OP CODE DESCRIPTION. ITE "intersects" (logically AND's) the active quarters of E with the content of the selected Memory Element register. The logical product is left in the E register. ITE is an indexable and configurable instruction.

SPECIAL COMMENT. The logical AND of the content of M and E is formed by copying the ZEROS of M into E.

DETAILS. The logic of this OP code is the same as that of LDE, except that ZEROS are copied into E under permuted activity control in $QK^{13\alpha}$, instead of ZEROS and ONES, and no sign extension occurs.



QK Time Level	Step	Mem	M	E	Operation
ITE ILLUSTRATIVE EXAMPLE					
—	—	y	m	e	
$QK^{02}-QK^{11}$	1		y	e	Read
$QK^{11\beta}$	2		y	$e_{\bar p}$	"Intersect" Configured Contents of Memory with Contents of E
$QK^{13\alpha}$	3		y	$e_{\bar p} \mid y_{ap} \cdot e_{\bar p}$	
$QK^{13\beta}$	4		y	$e \mid y_{CF} \cdot e$	
$QK^{21}-QK^{31}$	5	y	y	$e \mid y_{CF} \cdot e$	Rewrite

INTERSECT E (WITH MEMORY)

PK	24	α		$\lfloor 00 \rightarrow PK$

QK	00	α	QI^{start} . . . . $\supset$	$\lfloor start \rightarrow FK$, $\lfloor 0 \rightarrow PI_1$
	01	α		
			SEE QKM TIMING	
	09	α		
	10	α		
	11	α		$\lfloor 13 \rightarrow QK$
		β		$\lfloor \bar{p} \rightarrow E$
	13	α		$M \xrightarrow[\alpha,\beta]{0} E$
		β		$\lfloor p \rightarrow E$
	14	α		$\lfloor 21 \rightarrow QK$
	21	α		
	22	α		
	23	α		$\lfloor 0 \rightarrow EB$
			SEE QKM TIMING	
	31	α		

OP Class Decoder Lines Up:

$PKIR^{def}$
$PKIR^{ind}$
$PKIR^{QK}$
$QKIR^{load}$
$QKIR^{E}$

C.A.N 6-8-61

INTERSECT A (WITH MEMORY)

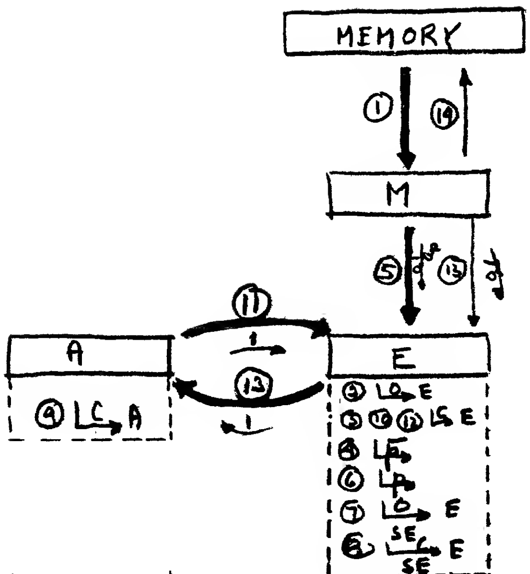
OP CODE DESCRIPTION. ITA "intersects" (logically AND's) the active subwords in A with the content of the selected Memory Element register. The logical product is placed in the A register. ITA is an indexable and configurable instruction.

SPECIAL COMMENT. The logical OR of the content of A and E is formed by copying the ONES of A into E. The logical AND of the two factors is formed by complementing in E the logical OR of their two complements.

DETAILS. The E register is cleared and complemented. The content of M is transferred into E under permuted activity control and then the content of E is directly permuted. Finally, the sign of the configured operand is extended. These operations set up E for the logical manipulations that take place during the balance of the QK cycle.

At $QK^{21\alpha}$ the inactive subwords of E contain ONES and the active subwords contain the configured operand with its sign extended. The A and E registers are now complemented. This places ZEROS in the inactive subwords of E.

The ONES in A are now transferred into E. This leaves the logical sum $(\bar{a} + \overline{y_{CF_{SE}}})$ in the active subwords of E and $\bar{a}$ in the inactive subwords of E. E is now complemented. The active subwords of E now contain the logical product $(y_{CF_{SE}} \cdot a)$ and the inactive subwords contain a. The content of E (the logical AND) is now copied into A. The original operand (y) is rewritten in memory and also copied into E.



QK Time Level	Step	Memory	M	E	A	Operation
—	—	y	m	e	a	
$QK^{10}-QK^{11}$	①		y	0	a	Read
$QK^{10\alpha}$	②		y	0	a	E Set Up
$QK^{10\beta}$	③		y	1	a	
$QK^{11\beta}$	④		y	$1\bar{p}$	a	Configuration
$QK^{13\alpha}$	⑤		y	$1_{\bar{p}} \mid y_{a_p}$	a	
$QK^{13\beta}$	⑥		y	$1 \mid y_{CF}$	a	
$QK^{14\alpha}$	⑦		y	$1 \mid y_{CF_{SE}}$	a	Sign Extension
$QK^{14\beta}$	⑧		y		a	
$QK^{21\alpha}$	⑨		y	$1 \mid y_{CF_{SE}}$	$\bar{a}$	"Intersect" Sign Extended-Configured Contents of Memory With Contents of A.
$QK^{21\beta}$	⑩		y	$0 \mid \overline{y_{CF_{SE}}}$	$\bar{a}$	
$QK^{22\alpha}$	⑪		y	$\bar{a} \mid \bar{a}+\bar{y}_{CF_{SE}}$	0	
$QK^{22\beta}$	⑫		y	$a \mid a \cdot y_{CF_{SE}}$	0	
$QK^{23\beta}$	⑬		y	y	$a \mid a \cdot y_{CF_{SE}}$	
$QK^{24}-QK^{31}$	⑭	y	y	y	$a \mid a \cdot y_{CF_{SE}}$	Rewrite

INTERSECT A (WITH MEMORY)

PK	24	α	$L100 \rightarrow PK$

QK	00	α	QI^{start} . . . . $\supset$ $L^{start} \rightarrow FK$, $L0 \rightarrow PI_1$
	01	α	
	SEE QKM TIMING		
	09	α	
	10	α	$L0 \rightarrow E$
		β	$Lc \rightarrow E$
	11	α	$L13 \rightarrow QK$
		β	$L\bar{P} \rightarrow E$
	13	α	$M \xrightarrow[a,p]{0,1} E$
		β	$LP \rightarrow E$
	14	α	$L0_{SE} \rightarrow E$ $L21 \rightarrow QK$
		β	$Lc_{SE} \rightarrow E$
	21	α	$Lc \rightarrow A$
		β	$Lc \rightarrow E$
	22	α	$A \xrightarrow{1} E$ $L0 \rightarrow A$
		β	$Lc \rightarrow E$
	23	α	$E \xrightarrow{1} A$ $M \xrightarrow{0,1} E$ $L0 \rightarrow EB$
	SEE QKM TIMING		
	31	α	

OP Class Decoder Lines Up:

$PKIR^{def}$
$PKIR^{ind}$
$PKIR^{QK}$
$PKIR^{AE}$
$QKIR^{ld}$
$QKIR^{load}$

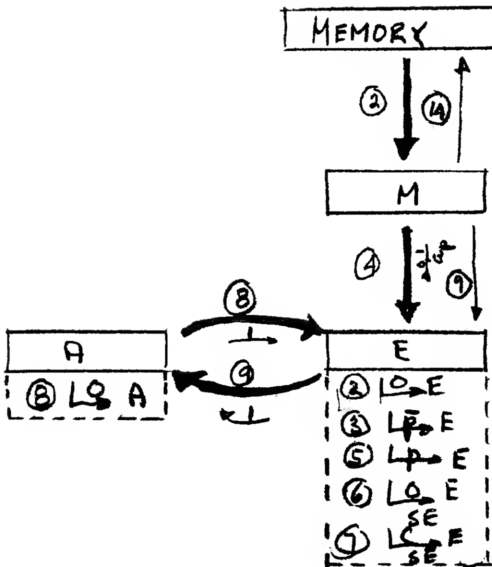
C.A.N. 6-8-61

UNITE A (WITH MEMORY)

OP CODE DESCRIPTION. UNA "unites" (logically OR's) the active subwords in A with the content of the selected Memory Element register. The logical sum is placed in the A register. UNA is an indexable and configurable instruction.

SPECIAL COMMENT. The logical OR of the content of the A and E register is formed by copying the ONES of A into E.

DETAILS. The execution logic for UNA is identical to that for ITA except for the three complement pulses to A and E at $QK^{21\alpha}$, $QK^{21\beta}$ and $QK^{22\beta}$. Thus the logical OR, rather than the logical AND of the two numbers is placed in A.



QK Time Level	Step	Memory	M	E	A	Operation
—	—	y	m	e	a	
QK^{02}–QK^{11}	1					Read
$QK^{10\alpha}$	2		y	0	a	Clear E
$QK^{11\beta}$	3		y	$0_{\bar{p}}$	a	Configuration
$QK^{13\alpha}$	4		y	$0 \mid y_{a_p}$	a	
$QK^{13\beta}$	5		y	$0 \mid y_{CF}$	a a	
$QK^{14\alpha}$	6		y	$0 \mid y_{CF_{SE}}$	a	Sign Extension
$QK^{14\beta}$	7		y			
$QK^{22\alpha}$	8		y	$a, \mid y_{CF_{SE}+a}$	0	"Unite" sign-extended, configured contents of memory with contents of A
$QI^{23\alpha}$	9		y	y	$a \mid y_{CF_{SE}+a}$	
QK^{21}–QK^{4}	10	y	y	y	$a \mid y_{CF_{SE}+a}$	Rewrite

UNITE A (WITH MEMORY)

PK	24	α		$\lfloor 00 \rightarrow$ PK

	00	α	QI^{start} . . . . ⊃	$\lfloor start \rightarrow$ FK, $\lfloor 0 \rightarrow PI_1$
	01	α		
	SEE QKM TIMING			
	09	α		
	10	α		$\lfloor 0 \rightarrow$ E
	11	α		$\lfloor 13 \rightarrow$ QK
		β		$\lfloor \bar{p} \rightarrow$ E
	13	α		$M \xrightarrow[\alpha, p]{0,1} E$
		β		$\lfloor p \rightarrow$ E
QK	14	α		$\lfloor 0_{SE} \rightarrow$ E $\lfloor 21 \rightarrow$ QK
		β		$\lfloor c_{SE} \rightarrow$ E
	21	α		
	22	α		$A \xrightarrow{1} E$ $\lfloor 0 \rightarrow$ A
	23	α		$E \xrightarrow{1} A$ $M \xrightarrow{0,1} E$ $\lfloor 0 \rightarrow$ EB
	SEE QKM TIMING			
	31	α		

OP Class Decoder Lines Up. $PKIR^{def}$
$PKIR^{ind}$
$PKIR^{QK}$
$PKIR^{AE}$
$QKIR^{ld}$
$QKIR^{load}$

SKIP IF E DIFFERS (FROM MEMORY)

OP CODE DESCRIPTION. SED compares the content of the E register with the content of the selected Memory Element register; if any of the active subwords "differ", a SKIP occurs, i.e., P is indexed twice during the PK cycle instead of once. SED is an indexable and configurable instruction.

SPECIAL FEATURES. SED has an extended PK cycle (PK^{25} through PK^{31}). SED is also characterized by: (1) double indexing of P; (2) "exclusive or" transfers between M and E under permuted activity control; and (3) a $PK^{25\alpha}$ waiting state. In this instruction, the active quarters of E are sampled for a non-zero condition by an $E^{SKIP\ \overline{ZERO}}$ net.

SED
43

DETAILS. The SED example shown was worked out for a specific configuration and for specific numerical values of operand and data in E. The general features of the instruction should be apparent from the example.

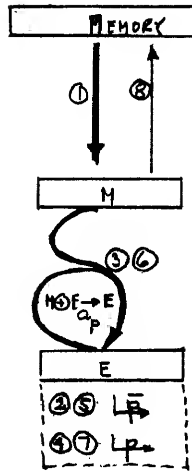
In the example, the original content of E is shifted quarter wise to the right by an inverse permutation pulse. An "exclusive or" transfer between M and E under permuted activity control then occurs. This is followed by a direct permutation. This process compares the bits in the active quarters of E with the corresponding bits of the configured operand. If the compared bits are identical, ZEROS are left in the corresponding E bit positions; if they are not identical, ONES are left in the E bit positions.

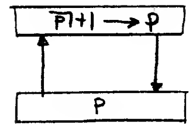
At $QK^{14\alpha}$, E_1 contains $y_4 + e_1$ and E_2 contains $y_1 + e_2$. In the numerical example, $y_4 = e_1$, therefore $y_4 + e_i$ is all ZEROS. However, $y_1 \neq e_2$, therefore $y_1 + e_2$ contains some ONES.

The $E^{SKIP\ \overline{ZERO}}$ net samples the active quarters of E. An "E different from memory" condition is discovered in E_2 and an $E^{SKIP\ \overline{ZERO}}$ level is generated. PK meanwhile jumps to the $PK^{31\alpha}$ state from the $PK^{25\alpha}$ waiting state. Since $PK^{31\alpha}$ sees an $E^{SKIP\ \overline{ZERO}}$ level, P is indexed (note that P was already indexed in $PK^{24\alpha}$).

Note that the change sequence condition are sampled both in PK^{24} and in PK^{31}.

The balance of the QK cycle restores E to its original content and executes the write cycle. The numerical example shows how the second "exclusive or" transfer restores E to its original value.





$E^{SKIP\ \overline{ZERO}} \cdot PK^{31\alpha} \supset \overline{P}+1 \rightarrow P$

(SEE BELOW)

SED ILLUSTRATIVE EXAMPLE

FRACTURE . . . . $f_4(9,9,9,9)$

PERMUTATION . . . . pam 3 (rotate one quarter to the left)

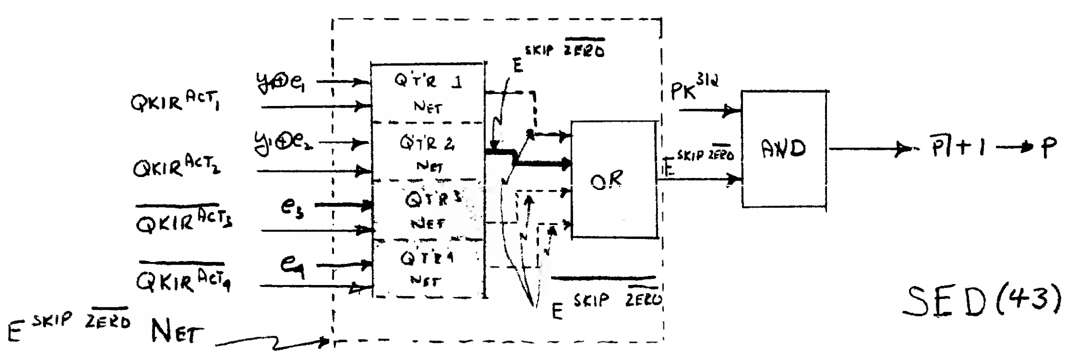
ACTIVITY . . . . . { QUARTERS 1 & 2 ACTIVE; " 3 & 4 INACTIVE

OPERAND & $\overline{E}$ . . . . SEE NUMERICAL EXAMPLE BELOW

QK	STEP	MEMORY	M	E				OPERATION
—	—	y	m	e				
02-11	①		y_4 y_3 y_2 y_1	e_4	e_3	e_2	e_1	READ
11β	②			e_1	e_4	e_3	e_2	"COMPARE" ACTIVE QUARTERS OF M and E. SAMPLE SKIP NET (See Below). SKIP IF E DIFFERS, I.E. IF ANY ACTIVE QUARTER OF E IS ZERO
13α	③			$y_1 \oplus e_1$	e_4	e_3	$y_1 \oplus e_2$	
13β	④			e_4	e_3	$y_1 \oplus e_2$	$y_4 \oplus e_1$	
21β	⑤			$y_4 \oplus e_1$	e_4	e_3	$y_1 \oplus e_2$	RESTORE E
23α	⑥			e_1*	e_4	e_3	e_2*	
23β	⑦		y	e_4	e_3	e_2	e_1	
21-31	⑧	y	y	e				REWRITE

NUMERICAL EXAMPLE OF EXCLUSIVE OR TRANSFER ($M \oplus E \rightarrow E$)

y_1	0 1 0 1 0 1 1 0 0	y_4	0 1 0 1 1 1 0 1 0
e_2	0 1 0 1 0 1 0 1 0	e_1	0 1 0 1 1 1 0 1 0
$y_1 \oplus e_2$	0 0 0 0 0 0 1 1 0	$y_4 \oplus e_1$	0 0 0 0 0 0 0 0 0
$y_1 \oplus (y_1 \oplus e_2) = e_2$	0 1 0 1 0 1 0 1 0	$y_4 \oplus (y_4 \oplus e_1) = e_1$	0 1 0 1 1 1 0 1 0



SED (43)

SKIP IF E DIFFERS (WITH MEMORY)

	24	α		
PK	25	α	$QKIR^{sed} \cdot QK^{14\alpha}$ . . . . . . $\supset$	$\overline{PK^7}+1 \not\longrightarrow PK$ $\lfloor 31 \rightarrow PK$
	31	α	$E^{skip\ \overline{zero}}$ . . . . $\supset$ $PI^{ch\ seq}$ . . . . $\supset$	$\overline{P^7}+1 \longrightarrow P$ $\lfloor 1 \rightarrow PI_3$

	00	α	QI^{start} . . . . $\supset$	$\lfloor start \rightarrow FK$, $\lfloor 0 \rightarrow PI_1$
	01	α		
	SEE QKM TIMING			
	09	α		
	10	α		
	11	α		$\lfloor 13 \rightarrow QK$
	11	β		$\lfloor \overline{P} \rightarrow E$
	13	α		$M \oplus E \xrightarrow[a,p]{} E$
QK	13	β		$\lfloor P \rightarrow E$
	14	α		$\lfloor 21 \rightarrow QK$ $\lfloor 31 \rightarrow PK$
	21	β		$\lfloor \overline{P} \rightarrow E$
	22	α		
	23	α		$M \oplus E \xrightarrow[a,p]{} E$
	23	β		$\lfloor 0 \rightarrow EB$
	SEE QKM TIMING			
	31	α		

OP Class Decoder Lines Up:

- $PKIR^{def}$
- $PKIR^{ind}$
- $PKIR^{QK}$
- $PKIR^{dis}$
- $QKIR^{load}$
- $QKIR^{E}$

$$E^{skip\ \overline{zero}} = \sum_{1} QKIR^{act_i} \cdot \left(\sum_{1}^{9} E^{1}_{i \cdot j}\right)$$

C.A.N. 6-8-61

JUMP ON OVERFLOW (IN A)

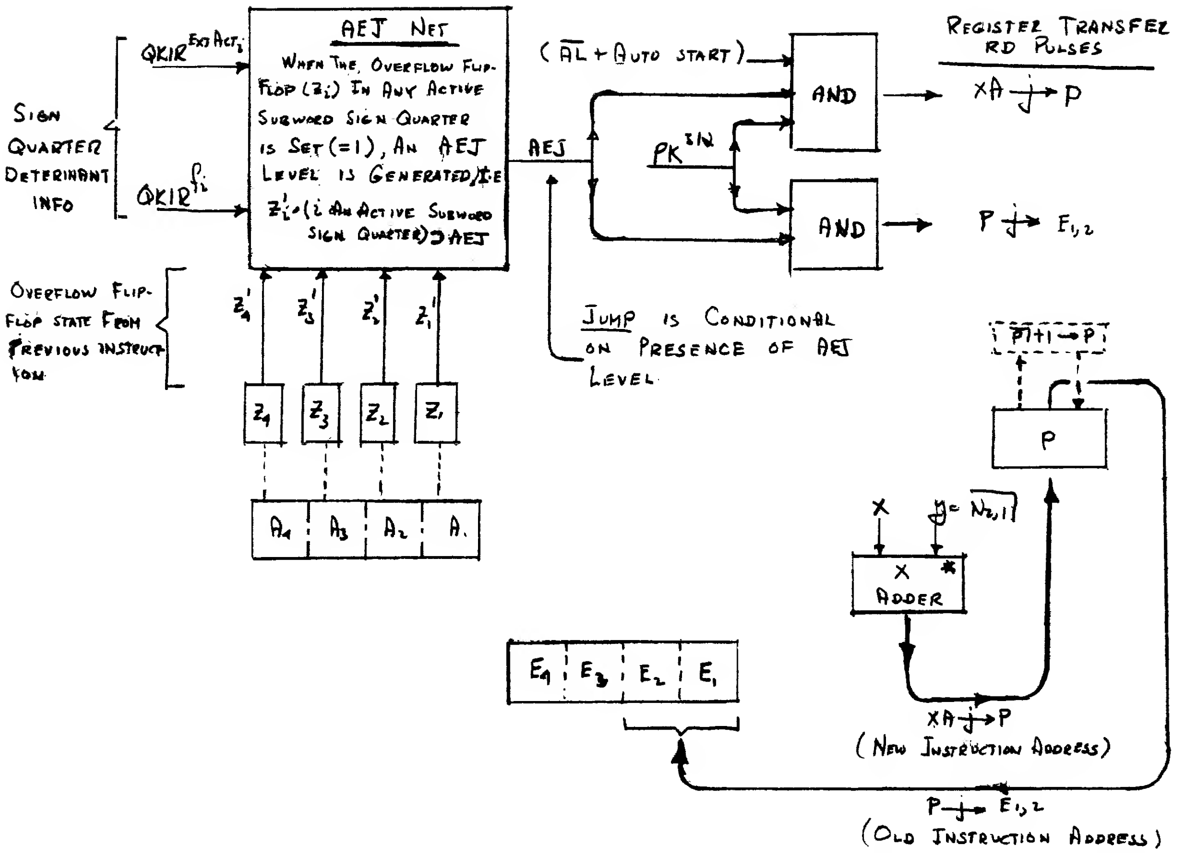
OP CODE DESCRIPTION. JOV performs a "jump" to the specified memory address, if the overflow flip-flop in the sign quarter of any active subword of A is set (Z_1^1). JOV is an indexable and configurable instruction.

SPECIAL FEATURES. JOV has an extended PK cycle (PK^{21} through PK^{31}) and no QK cycle. This is an instruction in which the FK counter is started in the PK cycle, since configuration information is used to determine the fracture and activity of the A register. An AEJ net is used to sample the Z overflow flip-flops.

DETAILS. $PK^{31\alpha}$ samples the AEJ net. If an AEJ level is present, the output of the X Adder is strobed into P and the content of P is transferred into $E_{2,1}$. The output of the X Adder is the indexed base address.

Note that P is not changed if an alarm condition exists (AL) unless the Auto Start switch is turned on.

Note also that the change sequence condition is sampled both in PK^{24} and PK^{31}.



* THE OUTPUT OF THE X ADDER (XA) EQUALS THE ARITHMETIC SUM OF X AND $N_{1,2}$

JUMP ON OVERFLOW (IN A)

24	α		
25	α	$AEB + QB' + FI^{\circ} \cdots \supset$	$\overline{PK'} + 1 \rightarrow PK$
26	α		$\lfloor 31 \rightarrow PK$
PK 31	α	$AEJ \cdots \supset$ $AEJ \cdot (\overline{AL} + \text{AUTO-START}) \cdot \supset$ $PI^{ch\ seq} \cdots \supset$	$P \xrightarrow{j} E_{2,1}$ $XA \xrightarrow{i} P$ $\lfloor 1 \rightarrow PI_3$

OP Class Decoder Lines Up:

$PKIR^{def}$
$PKIR^{ind}$
$PKIR^{AE}$
$PKIR^{ja}$
$PKIR^{dis}$

$$AEJ = PKIR^{jov} \cdot [\, Z_1' \cdot QKIR^{f3+f4} \cdot QKIR^{ext\,act_1} + Z_2' \cdot QKIR^{f2+f4} \cdot QKIR^{ext\,act_2} + Z_3' \cdot QKIR^{f4} \cdot QKIR^{ext\,act_3} + Z_4' \cdot QKIR^{ext\,act_4} \,]$$

C.A.N 6-8-61

JUMP ON POSITIVE A

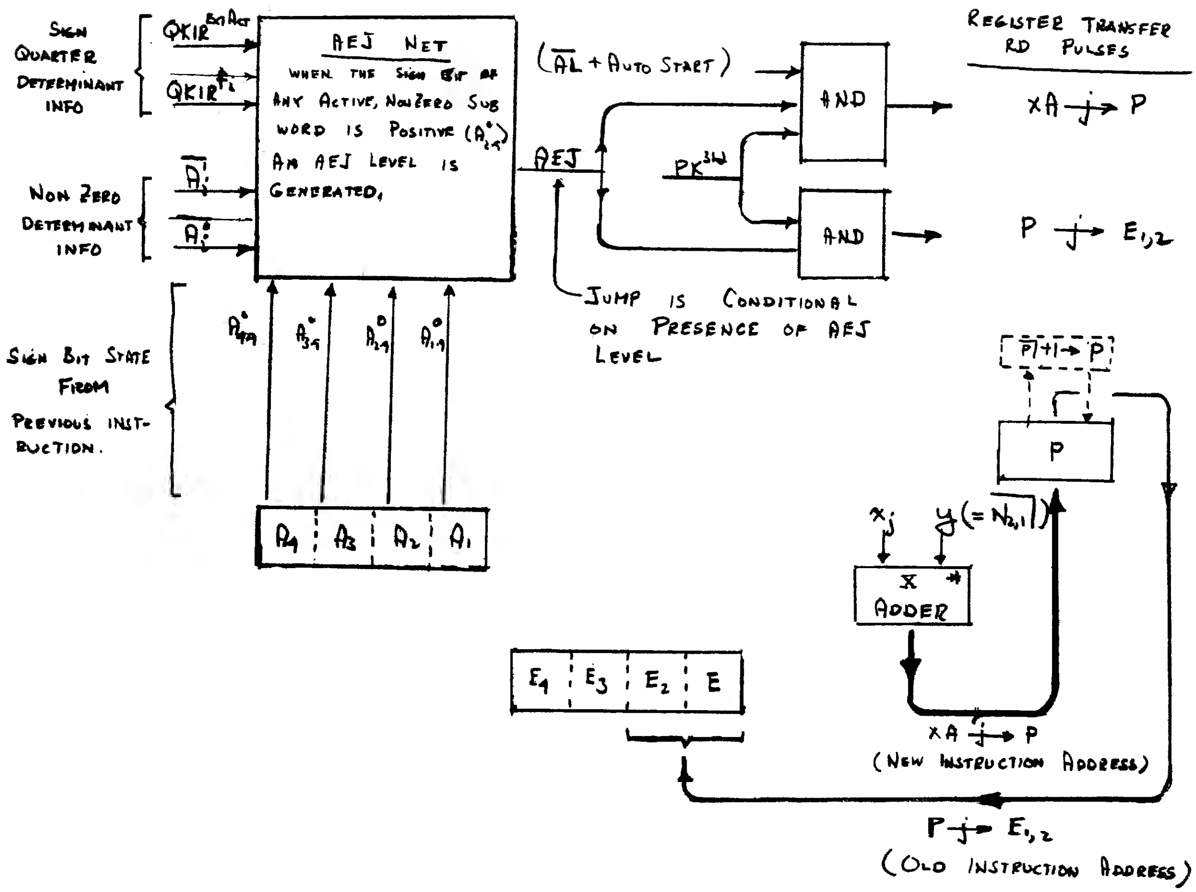
OP CODE DESCRIPTION. JPA performs a "jump" to the specified memory address, if the sign of any non-zero subword in A is positive. JPA is an indexable and configurable instruction.

SPECIAL FEATURES. JPA has an extended PK cycle (PK^{21} through PK^{31}) and no QK cycle. This is an instruction in which the FK counter is started in the PK cycle, since configuration information is used to determine the fracture and activity of the A register. An AEJ net is used to sample the state of the sign bits in the A register.

DETAILS. $PK^{31\alpha}$ samples the AEJ net. If an AEJ level is present, the output of the X Adder is strobed into P and the content of P is transferred into $E_{2,1}$. The output of the X Adder is the indexed base address.

Note that P is not changed if an alarm condition exists (AL) unless the AUTO START switch is turned on.

Note also that the change sequence condition is sampled both in PK^{24} and PK^{31}.



* THE OUTPUT OF THE Σ ADDER (ΣA) EQUALS THE ARITHMETIC SUM OF x_j AND y, i.e. $x_j + y$.

JUMP ON POSITIVE (IN A)

	24	α		
	25	α	AEB + QB$'$ + FI$^{\circ}$ . . . . . $\supset$	$\overline{PK^7} + 1 \xrightarrow{\ /\ } PK$
	26	α		$\lfloor 31 \rightarrow PK$
PK	31	α	AEJ . . . . $\supset$ AEJ . ($\overline{AL}$ + AUTO-START) . $\supset$ $PI^{ch\ seq}$ . . . . $\supset$	$P \xrightarrow{j} E_{2,1}$ $XA \xrightarrow{j} P$ $\lfloor 1 \rightarrow PI_3$

OP Class Decoder Lines Up: $PKIR^{def}$
$PKIR^{ind}$
$PKIR^{AE}$
$PKIR^{ja}$
$PKIR^{dis}$

$$\begin{aligned} AEJ = PKIR^{jpa} \cdot [\, & A_1^{\circ} \cdot QKIR^{f_3+f_4} \cdot \overline{A_1^{+0}} \cdot QKIR^{ext\,act_1} \\ + & A_2^{\circ} \cdot (QKIR^{f_2+f_4} \cdot \overline{A_2^{+0}} + QKIR^{f_2} \cdot \overline{A_1^{+0}}) \cdot QKIR^{ext\,act_2} \\ + & A_3^{\circ} \cdot QKIR^{f_4} \cdot \overline{A_3^{+0}} \cdot QKIR^{ext\,act_3} \\ + & A_4^{\circ} \cdot (\overline{A_4^{+0}} + \overline{QKIR^{f_4}} \cdot \overline{A_3^{+0}} + QKIR^{f_1+f_3} \cdot \overline{A_2^{+0}} + \\ & QKIR^{f_1} \cdot \overline{A_1^{+0}}) \cdot QKIR^{ext\,act_1} \,] \end{aligned}$$

C.A.N 6-9-61

JUMP ON NEGATIVE A

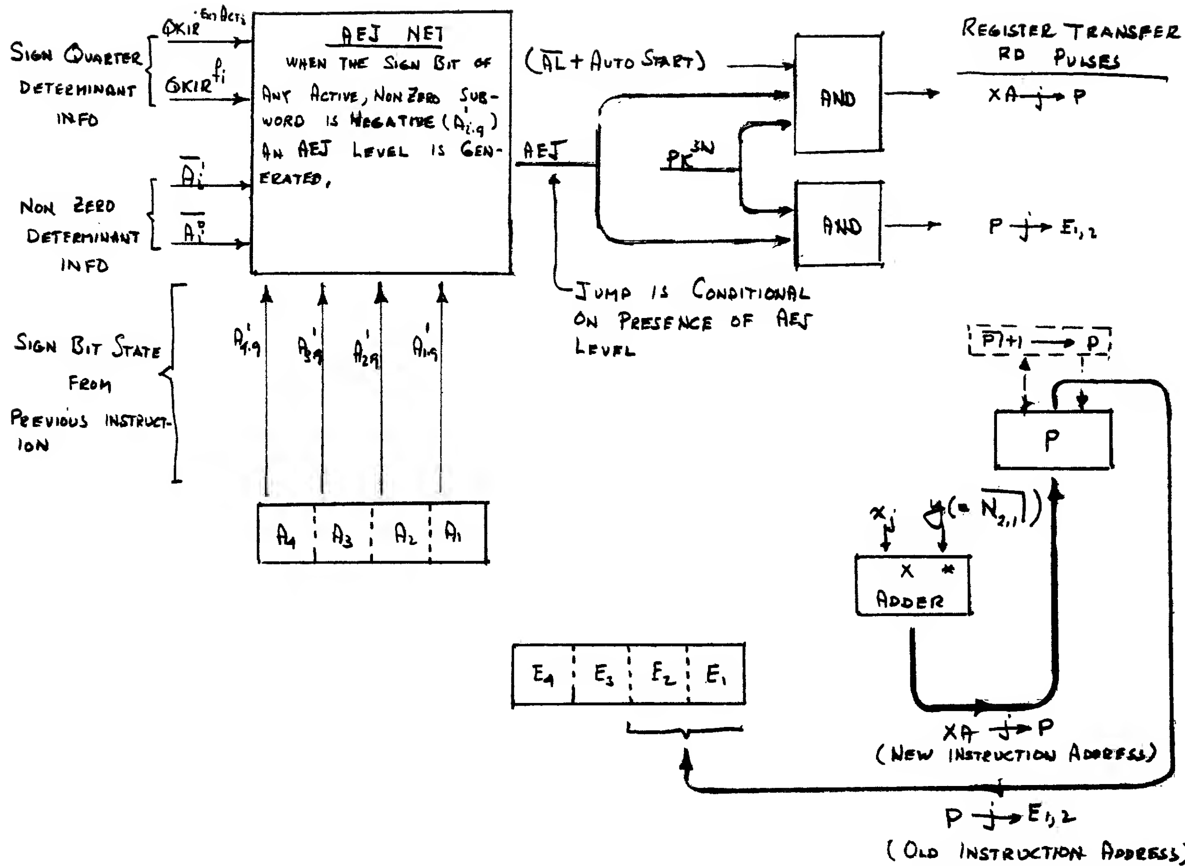
OP CODE DESCRIPTION. JNA performs a "jump" to the specified memory address, if the sign of any non-zero subword in A is negative. This is an indexable and configurable instruction.

SPECIAL FEATURES. JNA has an extended PK cycle (PK^{21} through PK^{31}) and no QK cycle. This is an instruction in which the FK counter is started in the PK cycle, since configuration information is used to determine fracture and activity in the A register. An AEJ net is used to sample the state of the sign bits in the A register.

DETAILS. $PK^{31\alpha}$ samples the AEJ net. If an AEJ level is present, the output of the X Adder is strobed into P and the content of P are transferred into $E_{2,1}$. The output of the X Adder is the indexed base address.

Note that P is not changed if an alarm condition exists (AL) unless the AUTO START switch is turned on.

Note also that the change sequence condition is sampled both in PK^{24} and PK^{31}.



* THE OUTPUT OF THE X ADDER (XA) EQUALS THE ARITHMETIC SUM OF x_j AND y

JUMP ON NEGATIVE (IN A)

24	α		
25	α	$AEB + QB^1 + FI^0 \cdot \cdot \cdot \cdot \cdot \supset$	$\overline{PK^1} + 1 \xrightarrow{/} PK$
26	α		$\lfloor 31 \rightarrow PK$
31	α	$AEJ \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \supset$ $AEJ \cdot (\overline{AL} + \text{AUTO-START}) \supset$ $PI^{ch\ seq} \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \supset$	$P \xrightarrow{j} E_{2,1}$ $XA \xrightarrow{j} P$ $\lfloor 1 \rightarrow PI_3$

PK

OP Class Decoder Lines Up:

$PKIR^{def}$
$PKIR^{ind}$
$PKIR^{AE}$
$PKIR^{ja}$
$PKIR^{dis}$

$$
\begin{aligned}
AEJ = PKIR^{jna} \cdot [\, & A_1^1 \cdot QKIR^{f_3+f_4} \cdot \overline{A_1^{-0}} \cdot QKIR^{ext\ act_1} \\
+ & A_2^1 \cdot (QKIR^{f_2+f_4} \cdot \overline{A_2^{-0}} + QKIR^{f_2} \cdot \overline{A_1^{-0}}) \cdot QKIR^{ext\ act_2} \\
+ & A_3^1 \cdot QKIR^{f_4} \cdot \overline{A_3^{-0}} \cdot QKIR^{ext\ act_3} \\
+ & A_4^1 \cdot (\overline{A_4^{-0}} + \overline{QKIR^{f_4}} \cdot \overline{A_3^{-0}} + QKIR^{f_1+f_3} \cdot \overline{A_2^{-0}} + QKIR^{f_1} \cdot \overline{A_1^{-0}}) \\
& \cdot QKIR^{ext\ act_1} \,]
\end{aligned}
$$

C.A.N 6-8-61

EXCHANGE A (WITH MEMORY)

OP CODE DESCRIPTION. EXA "exchanges" the content of the A register with the selected Memory Element register. EXA is a configurable and indexable instruction.

SPECIAL FEATURES. The content of M and E are "exchanged" under permuted activity control.

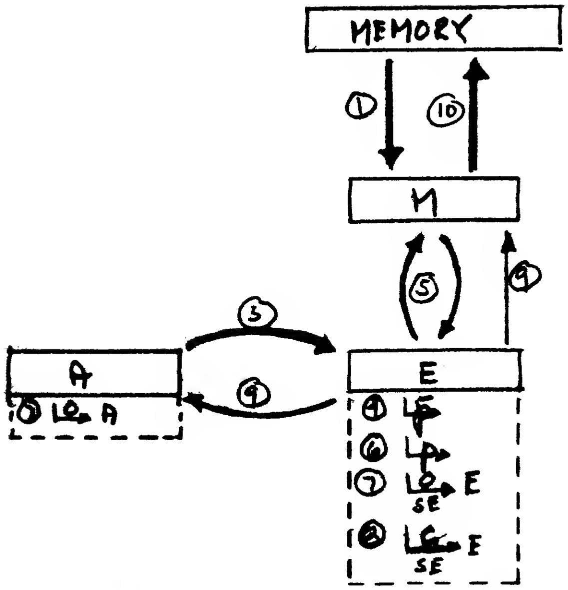
DETAILS. M is loaded with the operand word and E is loaded with the content of A.

The content of E is then inversely permuted; an interchange of the content of M and E under permuted activity control occurs; and then the content of E is directly permuted. This leaves the inversely configured content of A in M and the configured operand word in E. The sign of the configured operand word is then extended.

The balance of the QK cycle is used to load A with the configured operand word, with its sign extended, and write the configured original content of A in memory.

Note that this instruction essentially performs a LDA and a STA simultaneously.

EXA
54



QK	STEP	MEMORY	M	E	A	OPERATION
–	–	y	m	e	a	
01-11	1		y	e		READ
10α	2			0		E SET UP
11α	3			a		
~~11β~~	4			$a_{\bar{p}}$		CONFIGURATION
13α	5		$y \mid a_{CF}$	$a \mid y_{CF}$		
13β	6					
14α	7			$a \mid y_{CF_{SE}}$	0	SIGN EXTENSION
14β	8				0	
21α	9			$y \mid a_{CF}$	$a \mid y_{CF_{SE}}$	ULTIMATE PULSE
21-31	10	$y \mid a_{CF}$		$y \mid a_{CF}$	$a \mid y_{CF_{SE}}$	REWRITE

EXA (54)

EXCHANGE A (WITH MEMORY)

PK

24	α		$L00 \rightarrow PK$

QK

00	α	QI^{start} . . . . ⊃	$L^{start} \rightarrow FK_1$, $L0 \rightarrow PI_1$
01	α		
SEE QKM TIMING			
09	α		
10	α		$L0 \rightarrow E$
11	α		$A \xrightarrow{1} E$
	β		$L\bar{P} \rightarrow E$
12	α		
13	α	$\overline{MPA}$ . . . . ⊃	$M \xrightarrow[a,p]{0,1} E$ $E \xrightarrow[a,p]{0,1} M$
	β		$LP \rightarrow E$
14	α		$L0 \underset{SE}{\rightarrow} E$ $L0 \rightarrow A$ $L21 \rightarrow QK$
	β		$LC \underset{SE}{\rightarrow} E$
21	α		$E \xrightarrow{1} A$ $M \xrightarrow{0,1} E$
22	α		
23	α		$L0 \rightarrow EB$
SEE QKM TIMING			
31	α		

OP Class Decoder Lines Up:

$PKIR^{def}$	$QKIR^{A}$
$PKIR^{ind}$	$QKIR^{ld}$
$PKIR^{QK}$	$QKIR^{store}$
$PKIR^{AE}$	$QKIR^{st}$

C.A.N 6-9-61

INSERT (A IN MEMORY)

OP CODE DESCRIPTION. INS "inserts" (stores) the content of the flip-flops in A, corresponding to those flip-flops in B containing ONES, into the selected Memory Element register. The other memory bits in the selected Memory Element register are left unaffected. The effect of the instruction would be identical to that of a STA in which bits of A were transmitted to memory through a "mask" (or "sieve") corresponding to the ONES of B. INS is a configurable and indexable instruction.

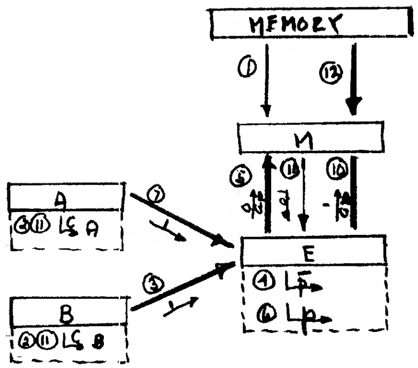
SPECIAL COMMENT. The logical AND of the content of two registers is formed by copying the ZEROS of one register into the second register. The logical OR of the content of two registers is formed by copying the ONES of one register into the second register. The logical AND of two factors is formed by complementing the OR of the complements of the two factors.

DETAILS. The timing and an example of the instruction are illustrated in the figure.

In the example the bits in A_3 corresponding to ONES in B_3 are placed in Y_2. Y_2 is associated with A_3 and B_3 because of the configuration. Whenever there are ZEROS in B_3, the corresponding Y_2 bits are left unaltered. The expression $(\overline{b}_3 \cdot y_2 + b_3 \cdot a_3)$ accomplishes the desired "masking" operation.

First $\overline{b}_3 \cdot y_2$ is formed in M_2. Then $b_3 \cdot a_3$ is formed in E_2. The logical OR of these two terms is formed by copying the ONES in E_2 into M_2. M_2 now contains $\overline{b}_3 \cdot y_2 \cdot b_3 \cdot a_3$ and this is rewritten in the Y_2 quarter of the memory register.

Similarly $(\overline{b}_2 \cdot y_1 + b_2 \cdot a_2)$ is formed and stored in the Y_1 quarter of the memory register.



INS ILLUSTRATIVE EXAMPLE

CONFIGURATION

DATA
See numerical example

$$INS = (\bar{b} \cdot y)_{CF} + (a \cdot f)_{CF} \longrightarrow Y$$

NUMERICAL EXAMPLE

y_2	1 0 0 1 1 0 0 0 1	y_1	1 0 1 0 1 0 1 0 1
b_3	1 1 0 0 1 1 1 0 0	b_2	0 0 0 0 0 1 1 1 1
a_3	0 1 1 0 0 1 0 1 1	a_2	0 1 0 1 0 1 0 1 0
$\bar{b}_3$	0 0 1 1 0 0 0 1 1	$\bar{b}_2$	1 1 1 1 1 0 0 0 0
$\bar{a}_3$	1 0 0 1 1 0 1 0 0	$\bar{a}_2$	1 0 1 0 1 0 1 0 1
$\bar{b}_3 \cdot y_2$	0 0 0 1 0 0 0 0 1	$\bar{b}_2 \cdot y_1$	1 0 1 0 1 0 0 0 0
$b_3 \cdot a_3$	0 1 0 0 0 1 0 0 0	$b_2 \cdot a_2$	0 0 0 0 0 1 0 1 0
$(\bar{b}_3 \cdot y_2 + b_3 \cdot a_3)$	0 1 0 1 0 1 0 0 1	$(\bar{b}_2 \cdot y_1 + b_2 \cdot a_2)$	1 0 1 0 1 1 0 1 0

QK	STEP	MEMORY	M				E				A				B				OPERATION
—	—	y	m				e				a				b				
02-11	①		y_4	y_3	y_2	y_1	e_4	e_3	e_2	e_1	a_4	a_3	a_2	a_1	b_4	b_3	b_2	b_1	READ
10α	②						0	0	0	0	$\bar{a}_4$	$\bar{a}_3$	$\bar{a}_2$	$\bar{a}_1$	$\bar{b}_4$	$\bar{b}_3$	$\bar{b}_2$	$\bar{b}_1$	E SET UP
11α	③						$\bar{b}_4$	$\bar{b}_3$	$\bar{b}_2$	$\bar{b}_1$									
11β	④						$\bar{b}_1$	$\bar{b}_4$	$\bar{b}_3$	$\bar{b}_2$									
13α	⑤				$\bar{b}_3 \cdot y_2$	$b_2 \cdot y_1$	$\bar{b}_1$	$\bar{b}_4$	$\bar{b}_3$	$\bar{b}_2$									
13β	⑥						$\bar{b}_4$	$\bar{b}_3$	$\bar{b}_2$	$\bar{b}_1$									LOGICAL TRANSFERS
14α	⑦						$\bar{b}_4+\bar{a}_4$	$\bar{b}_3+\bar{a}_3$	$\bar{b}_2+\bar{a}_2$	$\bar{b}_1+\bar{a}_1$									
14β	⑧						$b_4 \cdot a_4$	$b_3 \cdot a_3$	$b_2 \cdot a_2$	$b_1 \cdot a_1$									
18β	⑨						$b_1 \cdot a_1$	$b_4 \cdot a_4$	$b_3 \cdot a_3$	$b_2 \cdot a_2$									
19α	⑩				$\bar{b}_3 \cdot y_2 + b_3 \cdot a_3$	$\bar{b}_2 \cdot y_1 + b_2 \cdot a_2$	$b_1 \cdot a_1$	$b_4 \cdot a_4$	$b_3 \cdot a_3$	$b_2 \cdot a_2$									
21α	⑪						y_4	y_3	$\bar{b}_3 \cdot y_2 + b_3 \cdot a_3$	$\bar{b}_2 \cdot y_1 + b_2 \cdot a_2$	a_4	a_3	a_2	a_1	b_4	b_3	b_2	b_1	
21-31	⑫	y_4 | y_3 | $\bar{b}_3 \cdot y_2 + b_3 \cdot a_3$ | $\bar{b}_2 \cdot y_1 + b_2 \cdot a_2$																	REWRITE

INSERT (A IN MEMORY)

24	α		$\lfloor^{00} \rightarrow$ PK

00	α	QI^{start} . . . . ⊃	$\lfloor^{start} \rightarrow$ FK, $\lfloor^{0} \rightarrow PI_1$
01	α		
		SEE QKM TIMING	
09	α		
10	α		$\lfloor^{0} \rightarrow$ E $\lfloor^{c} \rightarrow$ A $\lfloor^{c} \rightarrow$ B
11	α		B $\xrightarrow{1}$ E
	β		$\lfloor^{\overline{P}} \rightarrow$ E
		SEE QKM TIMING	
13	α	$\overline{MPA}$ . . . . ⊃	E $\xrightarrow[\alpha,P]{0}$ M
	β		$\lfloor^{P} \rightarrow$ E
14 (QK)	α		A $\xrightarrow{1}$ E $\lfloor^{18} \rightarrow$ QK
		$MPAL_{sup}$ + $MPAL^{\circ}$ . . . . . . ⊃	$\lfloor^{1} \rightarrow$ MPS
	β		$\lfloor^{c} \rightarrow$ E
18	β		$\lfloor^{\overline{P}} \rightarrow$ E
19	α	$\overline{MPA}$ . . . . ⊃	E $\xrightarrow[\alpha,P]{1}$ M
20	α		
21	α		$\lfloor^{c} \rightarrow$ A $\lfloor^{c} \rightarrow$ B M $\xrightarrow{0,1}$ E
		$\overline{QKMVFF}$ . . . . ⊃	$\lfloor^{0} \rightarrow$ EB
22	α		
23	α		$\lfloor^{0} \rightarrow$ EB
		SEE QKM TIMING	
31	α		

OP Class Decoder Lines Up: $PKIR^{def}$, $PKIR^{ind}$, $PKIR^{QK}$, $PKIR^{AE}$, $QKIR^{B}$ & $QKIR^{store}$

C.A.N 6-9-61

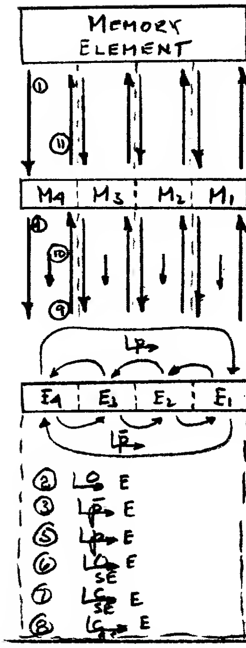
(PERMUTE AND) COMPLEMENT (MEMORY)

OP CODE DESCRIPTION. COM permutes the content of the selected Memory Element register and complements the active subwords. Sign extension also occurs in the active subwords. The result of the operation is placed both in E and in the selected Memory Element register. COM is an indexable and configurable instruction.

SPECIAL FEATURES. There are no transfers between M and E or E and M under permuted activity control. E is complemented under activity extension control.

DETAILS. E is cleared. The content of M, previously read out of memory, is then copied into E and a direct permutation pulse is fired off. Sign extension then occurs in the active subwords in E. The final step consists of complementing the active subwords of E.

The result now contained in E is transferred into M and written in the selected Memory Element register.



COM ILLUSTRATIVE EXAMPLE ( )											
QK	STEP	MEMORY	M				E				OPERATION
—	—	y	m_4	m_3	m_2	m_1	e_4	e_3	e_2	e_1	
62-11	①		y_4	y_3	y_2	y_1	↓	↓	↓	↓	READ
10α	②						0	0	0	0	CLEAR E
11β	③						0	0	0	0	PERMUTATION
13α	④						y_4	y_3	y_2	y_1	
13β	⑤						y_3	y_2	y_1	y_4	
14α	⑥						y_3	y_2	0	y_4	SIGN EXTENSION
14β	⑦						y_3	y_2	1*	y_4	
15β	⑧		↓	↓	↓	↓	y_3	y_2	0	$\bar{y}_4$	COMPLEMENT
16α	⑨		y_3	y_2	0	$\bar{y}_4$					LOAD M WITH E
21α	⑩										ULTIMATE PULSE
21-31	⑪	y_3 y_2 0 $\bar{y}_4$	↓	↓	↓	↓	↓	↓	↓	↓	REWRITE

* ASSUME $y_{4.9} = 1$

(PERMUTE AND) COMPLEMENT (MEMORY)

			Condition	Action
PK	24	α		$00 \rightarrow PK$
	00	α	QI^{start} . . . . $\supset$	$start \rightarrow FK$, $0 \rightarrow PI_1$
	01	α		
			SEE QKM TIMING	
	09			
	10	α		$0 \rightarrow E$
	11	α		$13 \rightarrow QK$
	11	β		$\bar{P} \rightarrow E$
	13	α		$M \xrightarrow{0,1} E$
	13	β		$P \rightarrow E$
	14	α		$0 \xrightarrow[SE]{} E$
	14	β		$C \xrightarrow[SE]{} E$
	15	β		$C \xrightarrow[ae]{} E$
	16	α	$\overline{MPA}$ . . . . $\supset$	$E \xrightarrow{0,1} M$
	17	α		$21 \rightarrow QK$
	21	α	$\overline{QKM^{VFF}}$ . . . . $\supset$	$M \xrightarrow{0,1} E$ $0 \rightarrow EB$
	22	α		
	23	α		$0 \rightarrow EB$
			SEE QKM TIMING	
	31	α		

OP Class Decoder Lines Up: $PKIR^{def}$, $PKIR^{ind}$, $PKIR^{QK}$, $QKIR^{store}$

C.A.N 6-9-61

TRANSFER DATA

(BETWEEN MEMORY AND IO BUFFER)

OP CODE DESCRIPTION. TSD transfers data between the specified IO Buffer and the selected Memory Element register. There are six different modes in which data can be transferred. TSD is an indexable and conditionally configurable instruction.

SPECIAL FEATURES. TSD has an extended PK cycle (PK^{25} through PK^{31}). The instruction is also characterized by: (1) TSD waiting state logic in $PK^{23\alpha}$ and $PK^{25\alpha}$; (2) cycle to the left and cycle to the right transfers from E to M; (3) IOCM control levels; (4) splayed data transfers; (5) no sign extension.

DETAILS. The IOCM levels ($IOCM^{OUT}$, $IOCM^{NORMAL}$, $IOCM^{LEFT}$) determine the six kinds of data transfer. $IOCM^{LEFT}$ is used only when the $\overline{IOCM^{NORMAL}}$ (i.e., $IOCM^{ASSEMBLY}$) level exists and affects the data transfer between E and M. $IOCM^{LEFT}$ ($\overline{IOCM^{LEFT}}$) indicates that the IO unit is running in the forward (reverse) direction. TSD data transfers are not affected when the $IOCM^{NORMAL}$ level exists. $IOCM^{OUT}$ determines whether data will be transferred from memory to the IO buffer, or vice versa.

NORMAL Data Transfer IN. The configured operand is placed in E without sign extension. The content of the IO Buffer, represented by IOBM, is then jammed into E. The IOBM levels can present either a ONE or ZERO input to a given bit of E, or neither. The content of E is then inversely configured and placed in M. The content of M is then written in memory, and copied into E.

NORMAL Data Transfer OUT. The configured operand is placed in E without sign extension and the IO Buffer is cleared. The content of that part of E which corresponds to the IO Buffer is then copied into the IO Buffer. This is done by the $\xrightarrow[KD]{DO}$ IOU pulse which occurs 0.8 microsecond after the E register is set up in order to allow signals on the IO Buffer to stabilize. The content of M is then written in memory, and copied into E.

ASSEMBLY Data Transfer IN (Forward/Reverse). In this case, the IO Buffer data is "assembled" rather than configured. The unconfigured 36 bit operand word is first copied into E. If a six bit IO Buffer is involved, every sixth bit in E is loaded with the content of a buffer bit. By means of six successive TSD's, 36 bits of input data (six lines) can be assembled in a single Memory Element register. During each TSD, the operand is rotated to the left one place if the IO unit is running in the forward direction ($IOCM^{LEFT}$) and to the right one place if the IO unit is running in the reverse direction ($\overline{IOCM^{LEFT}}$). This rotation occurs as the word is copied from E into M. The content of M is then written into memory and copied into M.

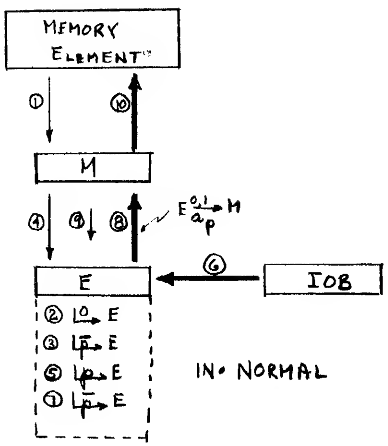
ASSEMBLY Data Transfer OUT (Forward/Reverse). The unconfigured 36 bit operand word is copied into E, and the IO Buffer is cleared. The content of the bits of E which correspond to the IO Buffer is then transferred to the IO Buffer. The bits selected depend on the $IOCM^{LEFT}$ level and the particular IO unit selected. In the case of a six bit IO Buffer, six successive output TSD's will disassemble a 36 bit memory word, so that six successive input TSD's can reassemble it. The content of E is cycled to the left (right) as it is copied back into M if the $IOCM^{LEFT}$ ($\overline{IOCM^{LEFT}}$) level exists. The content of M is then written in memory and at the same time copied into E.

Interlocking. There are several interlocking features that are peculiar to TSD. In addition to the ordinary wait conditions examined in $PK^{22\alpha}$, certain TSD wait conditions are examined. These wait conditions depend on whether the selected IO Buffer is busy or the QK cycle of a previous TSD is going on. If either of these conditions exist, the flag of the current sequence is lowered at PK^{22}. A change of sequence can then occur, or PK can wait in the $PK^{23\alpha}$ waiting state until the flag of the current sequence goes back up again or a change of sequence occurs.

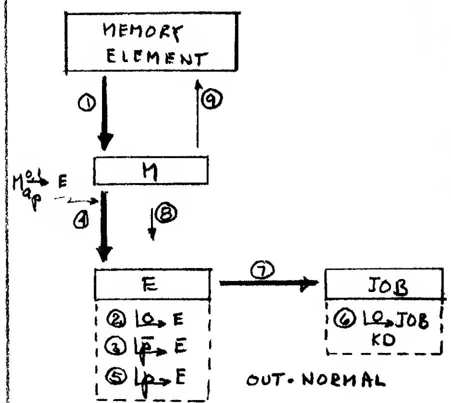
PK also waits in $PK^{25\alpha}$ for the QK cycle of the TSD to proceed past a certain state. The specific state depends on the hold bit of the TSD instruction. The hold bit is represented by the content of PI_4. If PI_4^1, then PK waits until $QK^{01\alpha}$, before jumping from $PK^{25\alpha}$ to $PK^{31\alpha}$. In this case the TSD will be followed by another instruction, i.e., the current PK cycle will be followed by another PK cycle.

If PI_4^0, then PK waits until $QK^{20\alpha}$ before jumping from $PK^{25\alpha}$ to $PK^{31\alpha}$. In this case the current sequence is dismissed ($PKIR_h^0 \cdot PKIR^{DIS\ REQ}$) and therefore the current PK cycle will be followed by a DSK or CSK cycle. Since either of these cycles would deselect the IO Buffer used by the TSD (by changing KD), PK is forced to wait in $PK^{25\alpha}$ (thus preventing the DSK or CSK cycle from occurring) until the QK cycle of the current TSD is essentially complete, i.e., until $QK^{20\alpha}$.

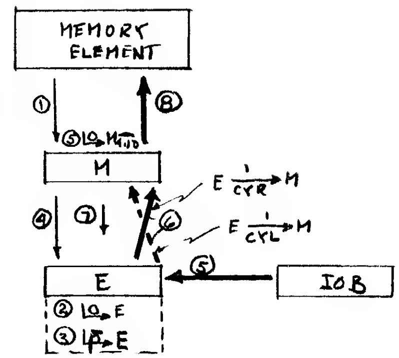
During the QK cycle of a TSD, no IOI clock pulses are allowed to be generated by an overlapping PK or DSK cycle, since these pulses can disturb the selected IO unit during the TSD.



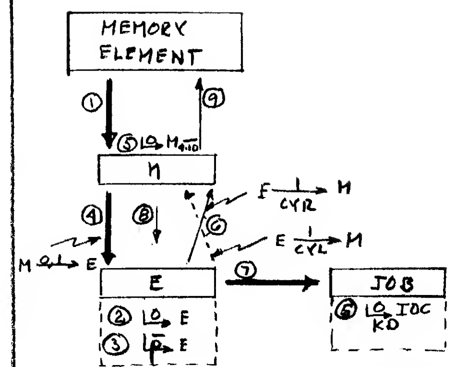
IN• NORMAL



OUT• NORMAL



IN · ASSY· RIGHT
(IN · ASSY· LEFT)



OUT· ASSY· RIGHT
(OUT• ASSY• LEFT)

TSD IN• NORMAL

QK	STEP	MEM	M	E	IOB
—	–	y	m	e	b
02-11	①		y	e	
10α	②			0	
11β	③			$0_{\bar p}$	
13α	④			y_{a_p}	
13β	⑤			y_{CF}	
18α	⑥			$y_{CF} \vert b$	
18β	⑦			$y_{a_p} \vert b_{\bar p}$	
19α	⑧		$y \vert b_{CF}$	$y_{a_p} \vert b_{\bar p}$	
21α	⑨			$y \vert b_{CF}$	
21-31	⑩	$y \vert b_{CF}$	↓	↓	↓

TSD OUT· NORMAL

QK	STEP	MEM	M	E	IOB
–	–	y	m	e	b
02-11	①		y	e	
10α	②			0	
11β	③			$0_{\bar p}$	
13α	④			y_{a_p}	
13β	⑤			y_{CF}	
18α	⑥				0
20α	⑦				y_{CF}
21α	⑧			y	
21-31	⑨	y	↓	↓	↓

TSD IN· ASSY· RIGHT

QK	STEP	MEM	M	E	IOB
—	–	y	m	e	b
02-11	①		y	e	
10α	②			0	
11β	③			$0_{\bar p}$	
13α	④			y	
18α	⑤		0	$y \vert b_{SP}$	
19α	⑥		$(y \vert b_{SP})_R$	$y \vert b_{SEl}$	
21α	⑦		$(y \vert b_{SP})_R$	$(y \vert b_{SP})_R$	
21-31	⑧	$(y \vert b_{SP})_R$	↓	↓	↓

TSD OUT· ASSY • RIGHT

QK	STEP	MEM	M	E	IOB
—	—	y	m	e	b
02-11	①		y	e	
10α	②			0	
11β	③			$0_{\bar p}$	
13α	④			y	
18α	⑤		0		0
19α	⑥		y_R		0
20α	⑦				y_{SP}
21α	⑧			y_R	
21-31	⑨	y_R	↓	↓	↓

TRANSFER DATA (BETWEEN MEMORY AND IO BUFFER)

PK	22	α	$PI_2^0 \cdot [IOCM^{BB} + (QB' \cdot QKIR^{tsd})] \supset$	$\lfloor \text{dismiss} \xrightarrow[KD]{} \diamond$ Flag $\lfloor 24 \rightarrow PK$
	24	α		
	25	α	$QKIR^{tsd} \cdot [(QK^{01\alpha} \cdot PI_4^1) + (QK^{20\alpha} \cdot PI_4^0)] \supset$ $PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{KD^{00}} \supset$	$\overline{PK'} + 1 \xrightarrow{/} PK$ $\lfloor 31 \rightarrow PK$ $\lfloor \text{dismiss} \xrightarrow[KD]{} \diamond$ Flag
	31	α	$PI^{ch\ req} \supset$ $PKIR_h^0 \cdot PKIR^{dis\ req} \cdot \overline{SS^{att\ req}} \supset$	$\lfloor 1 \rightarrow PI_3$ $\lfloor 1 \rightarrow CSK_4$
QK	00	α	$QI^{start} \supset$	$\lfloor \text{start} \rightarrow \diamond FK,\ \lfloor 0 \rightarrow PI_1$
	01	α	$PI_4^1 \supset$	$\lfloor 31 \rightarrow PK$
			SEE QKM TIMING	
	09	α		
	10	α		$\lfloor 0 \rightarrow E$
	11	α		$\lfloor 13 \rightarrow QK$
		β		$\lfloor \bar{P} \rightarrow E$
	13	α	$IOCM^{normal} \supset$ $IOCM^{assembly} \supset$	$\lfloor 18 \rightarrow QK$ $M \xrightarrow[\alpha,P]{0,1} E$ $M \xrightarrow{0,1} E$
		β	$IOCM^{normal} \supset$	$\lfloor P \rightarrow E$
	18	α	$IOCM^{in} \supset$ $IOCM^{out} \supset$ $IOCM^{assembly} \cdot \overline{MPA} \supset$	$IOBM \xrightarrow{\mid} E$ $\lfloor 0 \xrightarrow[KD]{} \diamond IOB$ $\lfloor 0 \rightarrow M_{4\text{-}10}$
		β	$IOCM^{in} \cdot IOCM^{normal} \supset$	$\lfloor \bar{P} \rightarrow E$
	19	α	$IOCM^{in} \cdot IOCM^{normal} \cdot \overline{MPA} \supset$ $IOCM^{assembly} \cdot IOCM^{left} \cdot (MPAL_{sup} + MPAL^0) \supset$ $IOCM^{assembly} \cdot IOCM^{right} \cdot (MPAL_{sup} + MPAL^0) \supset$	$E \xrightarrow[\alpha,P]{0,1} M$ $E \xrightarrow[cyl]{1} M$ $E \xrightarrow[cyr]{1} M$
	20	α	$PI_4^0 \supset$	$\lfloor DO \xrightarrow[KD]{} \diamond IOU$ $\lfloor 31 \rightarrow PK$
	21	α	$\overline{QKM^{VFF}} \supset$	$M \xrightarrow{0,1} E$ $\lfloor 0 \rightarrow EB$
	22	α		
	23	α		$\lfloor 0 \rightarrow EB$
			SEE QKM TIMING	
	31	α		

OP Class Decoder Lines Up: $PKIR^{def}$, $PKIR^{ind}$, $PKIR^{dis}$, $PKIR^{QK}$, $PKIR^{dis\ req}$, $QKIR^{store}$

$$IOB \xrightarrow[KD]{} \diamond E = QKIR^{tsd} \cdot EB'$$

16-6 PK-QK-AK INSTRUCTION CYCLES

16-6.1 INTRODUCTION

These instructions perform operations on data in the Arithmetic Element. They are characterized by the fact that the AK counter controls the pulses which occur in the Arithmetic Element.

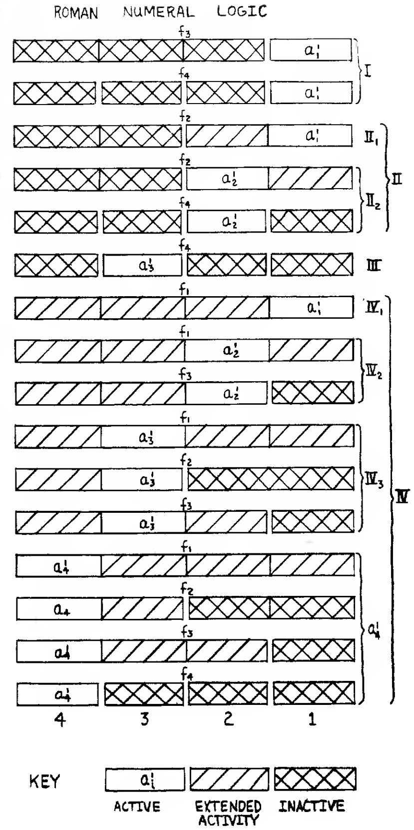
In all but one of these instructions an operand is first loaded into D. (In TLY, the operand is loaded into A.) Except for the pulse which transfers the contents of QKIR into AKIR in $QK^{13\alpha}$, and the pulses in $QK^{14\alpha}$ which start the AK counter and sets the "AE predict" (AEP) interlock, the PK-QK execution logic is identical to the LDD (LDA in the case of TLY) PK-QK execution logic. For this reason the timing charts in this section emphasize the events initiated by the AK counter.

Generally the illustrations accompanying the timing charts in this section give specific numerical examples. The state of each register involved in the execution of the instruction is given at each AK time state.

The figure on the next page tabulates the principal logic elements in the Arithmetic Element.

AE CONTROL LEVELS

OP CODE	INSTRUCTION		OCSAL	AOP	SH	SHA	SHB	CY	CY·AB	AB	CY·(A+B)	2N	NOR	ADD	ld load	A	D	AK	AESK
	AKIR OP & CLASS INSTRUCTION TABLE: $AKIR_{op}$														QKIR OP & CLASS INSTRUCTION TABLE: $QKIR_{op}$				
60	CYA				●	●		●	●		●				●		●	●	●
61	CYB				●		●	●	●		●				●		●	●	●
62	CAB				●	●	●	●	●	●	●	●			●		●	●	●
63	udf		●	●															
64	NOA										●		●		●		●	●	●
65	DSA										●				●		●	●	
66	NAB								●	●		●	●		●		●	●	●
67	ADD										●			●	●		●	●	
70	SCA				●	●					●				●		●	●	●
71	SCB				●		●				●				●		●	●	●
72	SAB				●	●	●		●	●		●			●		●	●	●
73	udf		●	●															
74	TLY							●	●						●	●		●	●
75	DIV							●	●	●					●		●	●	●
76	MUL								●	●					●		●	●	●
77	SUB								●					●	●		●	●	
00 – 57			●	●															
$AK^{1}_{\alpha.1}$			●	●															



$PKIR^{opr\,AE}$		$QKIR_{op}$ (60-77)				
($N_{2.6}$	+	$QKIR_{op_6}$)	=	$AKIR_{op_6}$		
($N_{2.5}$	+	$QKIR_{op_5}$)	=	$AKIR_{op_5}$		
($N_{2.4}$	+	$QKIR_{op_4}$)	=	$AKIR_{op_4}$		
($N_{2.3}$	+	$QKIR_{op_3}$)	=	$AKIR_{op_3}$		
($N_{2.2}$	+	$QKIR_{op_2}$)	=	$AKIR_{op_2}$		
($N_{2.1}$	+	$QKIR_{op_1}$)	=	$AKIR_{op_1}$		
($N_{1.9}$	+	$QKIR_{cf_9}$)	=	$AKIR_{cf_9}$		
($N_{1.8}$	+	$QKIR_{cf_8}$)	=	$AKIR_{cf_8}$		
($N_{1.7}$	+	$QKIR^{ext\,act_4}$)	=	$AKIR_{cf_7}$	=	a_4
($N_{1.6}$	+	$QKIR^{ext\,act_3}$)	=	$AKIR_{cf_6}$	=	a_3
($N_{1.5}$	+	$QKIR^{ext\,act_2}$)	=	$AKIR_{cf_5}$	=	a_2
($N_{1.4}$	+	$QKIR^{ext\,act_1}$)	=	$AKIR_{cf_4}$	=	a_1

$AEB = AK^{0}_{\alpha.0}$

$AEI = AEP' \cdot (QKIR^{AESK} \cdot \overline{QK^{00\alpha}} \cdot QK^{0}_{\alpha.5})$

$A_i^0 = A_{i.9}$ to $A_{i.1}$ CONTAINS ALL ZEROES

$A_i^1 = A_{i.9}$ to $A_{i.1}$ CONTAINS ALL ONES

$FD_i = D_{i.8}$ to $D_{i.1}$ CONTAINS ALL ONES

$LAD_i = D_{i.8}$ to $D_{i.4}$ CONTAINS ALL ONES

$N \dashrightarrow AKIR = PKIR^{opr\,AE} \cdot PK^{25\alpha}$

$N2Z_i = AKIR^{NOR} \cdot Z_i^1 \cdot AK^{1}_{\alpha.2}$

$Pad\ \phi_i = AKIR^{div} \cdot AK^{1}_{\beta.9} \cdot (A^{0}_{i.9} + ASK_7^1 + ASK_1^0)$

$Pad\ /_1 = AKIR^{mul} \cdot (AK^{1}_{\beta.2} + AK^{1}_{\beta.3})$

$Pad\ /_2 = [(AKIR^{dsa} + AKIR^{ADD}) \cdot AK^{1}_{\beta.2}] + (AKIR^{div} \cdot AK^{1}_{\beta.3})$

$QKIR \dashrightarrow AKIR = QKIR^{AK} \cdot QK^{13\alpha}$

$SAD_i = D_{i.7}$ to $D_{i.1}$ CONTAINS ALL ONES (NOT USED)

$\sigma_i = (A^{0}_{i.9} \cdot A^{0}_{i.8}) + (A^{1}_{i.9} \cdot A^{1}_{i.8})$

$\underline{start}\diamond AK = (PKIR^{opr\,AE} \cdot PK^{26\alpha}) + (QKIR^{AK} \cdot QK^{14\alpha})$

ASK PRESET TABLE					
CLASS LEVEL	f_1	f_2	$f_3 \cdot (a_4^1 + a_3^1 + a_2^1)$	$f_3 \cdot (a_4^0 \cdot a_3^0 \cdot a_2^0)$	f_1
$AKIR^N$	135	157	146	170	170
$AKIR^{2N}$	161	135	113	157	170

AE FRACTURE DECODING		
$AKIR_{cf_9}$	$AKIR_{cf_8}$	FRACTURE
0	0	f_1
0	1	f_2
1	0	f_3
1	1	f_4

$$
\begin{aligned}
AEJ = {} & \{PKIR^{jov} \cdot [(Z_4^1 \cdot QKIR^{ext\,act_4})] + [Z_3^1 \cdot QKIR^{ext\,act_3} \cdot f_4] + [Z_2^1 \cdot QKIR^{ext\,act_2} \cdot (f_1 + f_2)] + [Z_1^1 \cdot QKIR^{ext\,act_1} \cdot (f_3 + f_4)]\} \\
& + \Big\{\{[(PKIR^{jna} \cdot A^{1}_{4.9}) + (PKIR^{jpa} \cdot A^{0}_{4.9})]\} \cdot \{[\overline{A_4^1} + (\overline{A_3^1} \cdot \overline{f_4})] + [\overline{A_2^1} \cdot (f_1 + f_3)] + [\overline{A_1^1} \cdot f_1]\} \cdot \{\overline{A_4^0} + [\overline{A_3^0} \cdot \overline{f_4}] + [\overline{A_2^0} \cdot (f_1 + f_3)] + [\overline{A_1^0} \cdot f_1]\} \cdot \{QKIR^{ext\,act_4}\}\Big\} \\
& + \{[(PKIR^{jna} \cdot A^{1}_{3.9}) + (PKIR^{jpa} \cdot A^{0}_{3.9})] \cdot \overline{A_3^1} \cdot \overline{A_3^0} \cdot f_4 \cdot QKIR^{ext\,act_3}\} \\
& + [(PKIR^{jna} \cdot A^{1}_{2.9}) + (PKIR^{jpa} \cdot A^{0}_{2.9})] \cdot \{[\overline{A_2^1} \cdot \overline{A_2^0} \cdot (f_2 + f_4)] + [\overline{A_1^1} \cdot \overline{A_1^0} \cdot f_2]\} \cdot [QKIR^{ext\,act_2}] \\
& + [(PKIR^{jna} \cdot A^{1}_{1.9}) + (PKIR^{jpa} \cdot A^{0}_{1.9})] \cdot \overline{A_1^1} \cdot \overline{A_1^0} \cdot [f_3 + f_4] \cdot QKIR^{ext\,act_1}
\end{aligned}
$$

C.A.N 6-30-61

CYA 60

CYB 61

CAB 62

CYCLE A, B OR AB
SCALE A, B OR AB

OP CODE DESCRIPTION. In the CYcle OP codes the subwords in the A (B or AB) register are shifted bitwise to the left or right a number of places determined by the operand. The CYcle OP codes ignore the state of the overflow flip-flops and rotate the entire subwords.

The execution logic for the SCale OP codes is generally similar to that for the CYcle OP codes, except that the shifting is open ended and involves the overflow and sign bits.

SPECIAL FEATURES. The FD level indicates when the required shifting is completed, i.e., when the "count in D is finished". The Z flip-flops are not cleared in CYcle instructions, but are cleared in SCale instructions. These instructions use the Shift Coupling Units.

DETAILS. The clearing and presetting of ASK is an unnecessary operation since ASK levels are not used in the execution logic of either the CYcle or SCale OP codes. However, the ASK count pulses occur each time the subwords are shifted.

CYcle. D is loaded with the operand and the positive operand subwords complemented. The original sign of the subwords is remembered in the Y flip-flops. A shift left occurs if the sign is positive, and a shift right if negative.

The first pulse shifting the content of A and counting in D occurs at $AK^{03\alpha}$. The succeeding shift and count pulses occur in $AK^{04\alpha}$. The shift and count pulses continue until FD indicates that all the subwords are completely shifted.

Note that during these instructions the A and B coupling units connect the left end of the subwords to the right end of the same subwords so that the subwords are simply rotated, left or right, the specified number of places.

SCA 70

SCB 71

SAB 72

SCale. D is loaded with the operand and the positive operand subwords complemented. The logic for these instructions is identical to the logic for the CYcle instructions, with the following exceptions and additions:

The content of the sign digit in each subword is not altered, i.e., information can be shifted out of, but not into the sign digit position. If the shift is to the left, then the digit to the right of the sign digit is not shifted into the sign digit. Similarly, if the shift is to the right, then the right-most digit of the subword is not shifted into the sign digit position.

If there is an overflow left from a previous instruction, it is shifted into the subwords during SCA and SAB instructions and the overflow flip-flops in the sign digit position are cleared. If the shift is to the left (Y_i^0), then the sign digit is complemented before the shifting begins ($AK^{02\alpha}$). If the shift is to the right (Y_i^1), then the sign digit is complemented at the same time as the first shift and count pulses; this shifts a ONE (a ZERO in negative numbers) into the bit position to the right of the sign bit on the first shift.

CYA

QK		AK	Y_1	D_1	Z_1	A_1	OPERATION
14d	START	00d	x	xxxxxxxxx	1	000101010	
21d	E → D	01d	0	000000000	1	000101010	Load D
		02d	0	000000011	1	000101010	Complement D
		03d	0	111111100	1	000101010	Shift A & Count in D
		04d	0	111111101	1	001010100	$\overline{FD_1}$ Shift A & Count in D
		04d	0	111111110	1	010101000	$\overline{FD_1}$
		04d	0	111111111	1	101010000	FD_1
		00d	0	111111111	1	101010000	

CYA ILLUSTRATIVE EXAMPLE

OPERAND (LOADED IN D FROM MEMORY) . . . . . $\boxed{000000011}$

DATA (LEFT IN A FROM PREVIOUS INSTRUCTION) $\boxed{000101010}$

OVERFLOW STATE (LEFT IN Z FROM PREVIOUS INST.) . . $\boxed{1}$

CONFIGURATION

DERIVATIVE INFORMATION FOR THE CITED EXAMPLE

ROMAN NUMERAL (SIGN QUARTER) I

SUBSCRIPTED ROMAN NUMERALS I_1 (= I)
(ACTIVE QUARTERS WHICH HAVE RN FOR SIGN QUARTER)

CYA (60)

SCA

QK		AK	Y_1	D_1	Z_1	A_1	OPERATION
14d	START	00d	x	xxxxxxxxx	1	000101010	
21d	E → D	01d	0	000000000	1	000101010	Load D
		02d	0	000000011	1	000101010	Complement D; Complement A
		03d	0	111111100	1	1;00101010	Clear Z; Shift A & Count in D
		04d	0	111111101	0	1;01010101	$\overline{FD_1}$ Shift A & Count in D
		04d	0	111111110	0	1;1010101 1	$\overline{FD_1}$
		04d	0	111111111	0	1;1010101 11	FD_1
		00d	0	111111111	0	1;1010101 11	

SCA ILLUSTRATIVE EXAMPLE
(SAME DATA AND INSTRUCTION SPECIFICATION
AS CYA EXAMPLE ABOVE)

SCA (70)

CYA (60), CYB (61), CAB (62), SCA (70), SCB (71), SAB (72)

PK	24	α	$\xrightarrow{100}$ PK

CYCLE A	CYA	60
CYCLE B	CYB	61
CYCLE AB	CAB	62
SCALE A	SCA	70
SCALE B	SCB	71
SCALE AB	SAB	72

QK	00	α	QI^{start} . . . . ⊃ $\xrightarrow{start}$ FK, $\xrightarrow{0}$ PI_1
	01	α	
			SEE QKM TIMING
	09	α	
	10	α	$\xrightarrow{0}$ E
	11		$\xrightarrow{13}$ QK $D \xrightarrow{1} E$
		β	$\xrightarrow{\bar{P}}$ E
	13	α	$M \xrightarrow[\alpha_P]{0,1} E$, $\xrightarrow{0} AK_{11-1}$, $\xrightarrow{1} AK_0$ $QKIR_{cf_{9-8}} \xrightarrow{j} AKIR_{cf_{9-8}}$ $QKIR^{ext\ acts\ 4-1} \xrightarrow{j} AKIR_{cf_{7-4}}$ $QKIR_{op_{6-1}} \xrightarrow{j} AKIR_{op_{6-1}}$
		β	$\xrightarrow{P}$ E
	14	α	$\xrightarrow{21}$ QK $\xrightarrow{1}$ AEP $\xrightarrow[SE]{0}$ E $\xrightarrow{0}$ D $\xrightarrow{start}$ AK
		β	$\xrightarrow[SE]{C}$ E
	21	α	$E \xrightarrow{1} D$
	22	α	
	23	α	$M \xrightarrow{0,1} E$ $\xrightarrow{0}$ EB $\xrightarrow{31}$ QK
	31	α	

OP Class Decoder Lines Up: $QKIR^{AK}$, $QKIR^{AESK}$, $QKIR^{load}$, $QKIR^{ld}$ $QKIR^{D}$

OPERATE (ARITHMETIC ELEMENT: $N^0_{28} \cdot N^1_{27}$) OPR^{AE} 04

PK	24	α	
	25	α	QB' + AEB . . . . ⊃ $\overline{PK'}+1 \rightarrow$ PK AK^1_0 . . . . ⊃ $\xrightarrow{0} AK_{11-1}$, $\xrightarrow{1} AK_0$ " . . . . ⊃ $N_{2.6-2.1} \xrightarrow{j} AKIR_{op_{6-1}}$ " . . . . ⊃ $N_{1.9-1.4} \xrightarrow{j} AKIR_{cf_{9-4}}$ (NOTE: AEB = AK^0_0)
	26	α	$\xrightarrow{31}$ PK $\xrightarrow{1}$ AEP $\xrightarrow{start}$ AK
	31	α	$PI^{ch\ seq}$ . . . . ⊃ $\xrightarrow{1} PI_3$

OP Class Decoder Lines Up: $PKIR^{def}$, $PKIR^{dis}$, $PKIR^{AE}$

AK	00	α				$\xrightarrow{start}$ AK . . . . . . ⊃ $\overline{AK'}+1 \rightarrow$ AK $\xrightarrow{start}$ AK . . . . . . ⊃ $\xrightarrow{0}$ ASK
	01	α				$\xrightarrow{preset}$ ASK
			4TH QUARTER	3RD QUARTER	2ND QUARTER	1ST QUARTER
	02	α	$AKIR^{SH} \cdot Y^0_4 \cdot IV$ . . . ⊃ $\xrightarrow{C} D_4$ $(AKIR^{SCA} + AKIR^{SAB}) \cdot Z^1_4 \cdot Y^0_4 \cdot IV$ ⊃ $\xrightarrow{C} A_{4.9}$	$AKIR^{SH} \cdot Y^0_3 \cdot III$ . . . . . . ⊃ $\xrightarrow{C} D_3$ $(AKIR^{SCA} + AKIR^{SAB}) \cdot Z^1_3 \cdot Y^0_3 \cdot III$ ⊃ $\xrightarrow{C} A_{3.9}$	$AKIR^{SH} \cdot Y^0_2 \cdot II$ . . . . ⊃ $\xrightarrow{C} D_2$ $(AKIR^{SCA} + AKIR^{SAB}) \cdot Z^1_2 \cdot Y^0_2 \cdot II$ ⊃ $\xrightarrow{C} A_{2.9}$	$AKIR^{SH} \cdot Y^0_1 \cdot I$ . . . . ⊃ $\xrightarrow{C} D_1$ $(AKIR^{SCA} + AKIR^{SAB}) \cdot Z^1_1 \cdot Y^0_1 \cdot I$ ⊃ $\xrightarrow{C} A_{1.9}$
	03	α	$AKIR^{SH} \cdot \overline{FD_4} \cdot IV$ . . . ⊃ $\overline{D'_4}+1 \rightarrow D_4$ $(AKIR^{SCA} + AKIR^{SAB}) \cdot IV$ . . . ⊃ $\xrightarrow{0} Z_4$ $(AKIR^{SCA} + AKIR^{SAB}) \cdot Z^1_4 \cdot Y^1_4 \cdot IV$ ⊃ $\xrightarrow{C} A_{4.9}$ $AKIR^{SHA} \cdot \overline{FD_4} \cdot Y^0_4 \cdot a^1_4$ . . ⊃ $\xrightarrow{SHL} A_4$ $AKIR^{SHB} \cdot \overline{FD_4} \cdot Y^0_4 \cdot a^1_4$ . . ⊃ $\xrightarrow{SHL} B_4$ $AKIR^{SHA} \cdot \overline{FD_4} \cdot Y^1_4 \cdot a^1_4$ . . ⊃ $\xrightarrow{SHR} A_4$ $AKIR^{SHB} \cdot \overline{FD_4} \cdot Y^1_4 \cdot a^1_4$ . . ⊃ $\xrightarrow{SHR} B_4$	$AKIR^{SH} \cdot \overline{FD_3} \cdot III$ . . . . . . ⊃ $\overline{D'_3}+1 \rightarrow D_3$ $(AKIR^{SCA} + AKIR^{SAB}) \cdot III$ . . . . ⊃ $\xrightarrow{0} Z_3$ $(AKIR^{SCA} + AKIR^{SAB}) \cdot Z^1_3 \cdot Y^1_4 \cdot III$ ⊃ $\xrightarrow{C} A_{3.9}$ $AKIR^{SHA} \cdot [(\overline{FD_3} \cdot Y^0_3 \cdot III) + (\overline{FD_4} \cdot Y^0_4 \cdot IV_3)]$ ⊃ $\xrightarrow{SHL} A_3$ $AKIR^{SHB} \cdot [$ " " $]$ ⊃ $\xrightarrow{SHL} B_3$ $AKIR^{SHA} \cdot [(\overline{FD_3} \cdot Y^1_3 \cdot III) + (\overline{FD_4} \cdot Y^1_4 \cdot IV_3)]$ ⊃ $\xrightarrow{SHR} A_3$ $AKIR^{SHB} \cdot [$ " " $]$ ⊃ $\xrightarrow{SHR} B_3$	$AKIR^{SH} \cdot \overline{FD_2} \cdot II$ . . . . . ⊃ $\overline{D'_2}+1 \rightarrow D_2$ $(AKIR^{SCA} + AKIR^{SAB}) \cdot II$ . . . ⊃ $\xrightarrow{0} Z_2$ $(AKIR^{SCA} + AKIR^{SAB}) \cdot Z^1_2 \cdot Y^1_2 \cdot II$ ⊃ $\xrightarrow{C} A_{2.9}$ $AKIR^{SHA} \cdot [(\overline{FD_2} \cdot Y^0_2 \cdot II_2) + (\overline{FD_4} \cdot Y^0_4 \cdot IV_2)]$ ⊃ $\xrightarrow{SHL} A_2$ $AKIR^{SHB} \cdot [$ " " $]$ ⊃ $\xrightarrow{SHL} B_2$ $AKIR^{SHA} \cdot [(\overline{FD_2} \cdot Y^1_2 \cdot II_2) + (\overline{FD_4} \cdot Y^1_4 \cdot IV_2)]$ ⊃ $\xrightarrow{SHR} A_2$ $AKIR^{SHB} \cdot [$ " " $]$ ⊃ $\xrightarrow{SHR} B_2$	$AKIR^{SH} \cdot \overline{FD_1} \cdot I$ . . . . ⊃ $\overline{D'_1}+1 \rightarrow D_1$ $(AKIR^{SCA} + AKIR^{SAB}) \cdot I$ . . . . . ⊃ $\xrightarrow{0} Z_1$ $(AKIR^{SCA} + AKIR^{SAB}) \cdot Z^1_1 \cdot Y^1_1 \cdot I$ ⊃ $\xrightarrow{C} A_{1.9}$ $AKIR^{SHA} \cdot [(\overline{FD_1} \cdot Y^0_1 \cdot I) + (\overline{FD_4} \cdot Y^0_4 \cdot IV_1) + (\overline{FD_2} \cdot Y^0_2 \cdot II_1)]$ ⊃ $\xrightarrow{SHL} A_1$ $AKIR^{SHB} \cdot [$ " " " $]$ ⊃ $\xrightarrow{SHL} B_1$ $AKIR^{SHA} \cdot [(\overline{FD_1} \cdot Y^1_1 \cdot I) + (\overline{FD_4} \cdot Y^1_4 \cdot IV_1) + (\overline{FD_2} \cdot Y^1_2 \cdot II_1)]$ ⊃ $\xrightarrow{SHR} A_1$ $AKIR^{SHB} \cdot [$ " " " $]$ ⊃ $\xrightarrow{SHR} B_1$
	04	α	$AKIR^{SH} \cdot \overline{FD_4} \cdot IV$ . . ⊃ $\overline{D'_4}+1 \rightarrow D_4$ $AKIR^{SHA} \cdot \overline{FD_4} \cdot Y^0_4 \cdot a^1_4$ ⊃ $\xrightarrow{SHL} A_4$ $AKIR^{SHB} \cdot \overline{FD_4} \cdot Y^0_4 \cdot a^1_4$ ⊃ $\xrightarrow{SHL} B_4$ $AKIR^{SHA} \cdot \overline{FD_4} \cdot Y^1_4 \cdot a^1_4$ ⊃ $\xrightarrow{SHR} A_4$ $AKIR^{SHB} \cdot \overline{FD_4} \cdot Y^1_4 \cdot a^1_4$ ⊃ $\xrightarrow{SHR} B_4$	$AKIR^{SH} \cdot \overline{FD_3} \cdot III$ . . . . . ⊃ $\overline{D'_3}+1 \rightarrow D_3$ $AKIR^{SHA} \cdot [(\overline{FD_3} \cdot Y^0_3 \cdot III) + (\overline{FD_4} \cdot Y^0_4 \cdot IV_3)]$ ⊃ $\xrightarrow{SHL} A_3$ $AKIR^{SHB} \cdot [$ " " $]$ ⊃ $\xrightarrow{SHL} B_3$ $AKIR^{SHA} \cdot [(\overline{FD_3} \cdot Y^1_3 \cdot III) + (\overline{FD_4} \cdot Y^1_4 \cdot IV_3)]$ ⊃ $\xrightarrow{SHR} A_3$ $AKIR^{SHB} \cdot [$ " " $]$ ⊃ $\xrightarrow{SHR} B_3$	$AKIR^{SH} \cdot \overline{FD_2} \cdot II$ . . . . . . ⊃ $\overline{D'_2}+1 \rightarrow D_2$ $AKIR^{SHA} \cdot [(\overline{FD_2} \cdot Y^0_2 \cdot II_2) + (\overline{FD_4} \cdot Y^0_4 \cdot IV_2)]$ ⊃ $\xrightarrow{SHL} A_2$ $AKIR^{SHB} \cdot [$ " " $]$ ⊃ $\xrightarrow{SHL} B_2$ $AKIR^{SHA} \cdot [(\overline{FD_2} \cdot Y^1_2 \cdot II_2) + (\overline{FD_4} \cdot Y^1_4 \cdot IV_2)]$ ⊃ $\xrightarrow{SHR} A_2$ $AKIR^{SHB} \cdot [$ " " $]$ ⊃ $\xrightarrow{SHR} B_2$	$AKIR^{SH} \cdot \overline{FD_1} \cdot I$ . . . . ⊃ $\overline{D'_1}+1 \rightarrow D_1$ $AKIR^{SHA} \cdot [(\overline{FD_1} \cdot Y^0_1 \cdot I) + (\overline{FD_4} \cdot Y^0_4 \cdot IV_1) + (\overline{FD_2} \cdot Y^0_2 \cdot II_1)]$ ⊃ $\xrightarrow{SHL} A_1$ $AKIR^{SHB} \cdot [$ " " " $]$ ⊃ $\xrightarrow{SHL} B_1$ $AKIR^{SHA} \cdot [(\overline{FD_1} \cdot Y^1_1 \cdot I) + (\overline{FD_4} \cdot Y^1_4 \cdot IV_1) + (\overline{FD_2} \cdot Y^1_2 \cdot II_1)]$ ⊃ $\xrightarrow{SHR} A_1$ $AKIR^{SHB} \cdot [$ " " " $]$ ⊃ $\xrightarrow{SHR} B_1$
			$(FD_4 + \overline{IV})$ • $(LAD_4 + \overline{IV})$ •	$(FD_3 + \overline{III})$ • $(LAD_3 + \overline{III})$ •	$(FD_2 + \overline{II})$ • $(LAD_2 + \overline{II})$ •	$(FD_1 + \overline{I})$ ⊃ $\xrightarrow{0} AK_{11-r_0}$ $\xrightarrow{1} AK_0$ $(LAD_1 + \overline{I})$ ⊃ $\xrightarrow{0}$ AEP $\overline{ASK'}+1 \rightarrow$ ASK $AK+1 \nrightarrow$ AK
	00	α				

$LAD_i = D^1_{i.8-4}$
$FD_i = D^1_{i.8-1}$

$AKIR^{SH}$ = CYA (60) + CYB (61) + CAB (62) + SCA (70) + SCB (71) + SAB (72)
$AKIR^{SHA}$ = CYA (60) + CYB (61) + SCA (62) + SAB (72)
$AKIR^{SHB}$ = CYB (61) + CAB (62) + SCB (71) + SAB (72)

AKIR OP CLASS LEVELS UP: $AKIR^{SHA}$, $AKIR^{CY}$, $AKIR^{AB}$, $AKIR^{CY \cdot AB}$, $AKIR^{A+B}$, $AKIR^{N}$, $AKIR^{2N}$

ASK PRESET TABLE

CLASS LEVEL	f_1	f_2	$f_3 \cdot (a^1_4 + a^1_3 + a^1_2)$	$f_3 \cdot (a^0_4 \cdot a^0_3 \cdot a^0_2)$	f_4
$AKIR^{N}$	135	157	146	170	170
$AKIR^{2N}$	161	135	113	157	170

C.A.N 6-16-61

NORMALIZE A, AB

OP CODE DESCRIPTION. The active subwords in A (AB) are shifted to the left until $A_{i.9} \neq A_{i.8}$. The number of shifts required to accomplish this is subtracted from the content of the sign quarters of the operand subwords in D. If an overflow exists in an active subword, the subword is shifted one place to the right, the overflow is shifted into the sign bit, and the sign quarter of the operand subword in D is indexed.

SPECIAL FEATURES. A σ (sigma) level is used to indicate when $A_{i.9} = A_{i.8}$ in the sign quarter. If the data contains all ZEROS or all ONES, ASK prevents the number of shifts from exceeding the register length. These instructions use the Shift Coupling Units.

NOA 64

DETAILS. The case where an overflow has occurred in a previous instruction is shown in one of the accompanying examples. ASK is cleared and preset and D is loaded with the operand.

Since an overflow condition is indicated, the sign bit is complemented and at the same time the content of the data register is shifted to the right. Since a shift to the right occurs, ONE is added to the uncomplemented operand in D. The pulses doing all this are fired off in $AK^{02\alpha}$.

NAB 66

Pulses that complement D and clear the overflow flip-flops occur at $AK^{03\alpha}$.

The $\overline{\sigma}$ ($\overline{\text{sigma}}$) level seen at $AK^{04\alpha}$ prevents further shifting from occurring. D is complemented again and AK reset to $AK^{00\alpha}$ by this logic.

The other example shows the "no overflow" case. In this case the shifting is to the left and all the shifting and counting pulses occur in $AK^{04\alpha}$, after D has been complemented at $AK^{03\alpha}$. When a $\overline{\sigma}$ level occurs, the counting is inhibited, D is again complemented and AK is reset to $AK^{00\alpha}$.

If no $\overline{\sigma}$ level occurs, i.e., if the number being normalized is positive or negative zero, then ASK will eventually become positive and stop the shifting process.

NOA – ASSUMING OVERFLOW OCCURRED IN PREVIOUS INSTRUCTION								
QK		AK	ASK	Y_1	D_1	Z_1	A_1	OPERATION
14d	START	00d	X X X X X X X	X	X X X X X X X X X	1	0:0 0 1 0 1 0 1 0	CLEAR ASK
21d	$E^1 \to D$	01d	0 0 0 0 0 0 0	0	0 0 0 0 0 0 0 0 0	1	0:0 0 1 0 1 0 1 0	PRESET ASK; LOAD D
		02d	1 1 1 1 0 0 0	0	0 0 0 0 0 0 0 1 1	1	0:0 0 1 0 1 0 1 0	SHIFT RIGHT & COUNT IN D
		03d	1 1 1 1 0 0 0	0	0 0 0 0 0 0 1 0 0	1	1:0 0 0 1 0 1 0 1	COMPL D; CLEAR Z
		04d	1 1 1 1 0 0 0	0	1 1 1 1 1 1 0 1 1	0	1:0 0 0 1 0 1 0 1	$\overline{G}_1$ $\overline{D}$+1→D; 1C → D
		00d	1 1 1 1 0 0 1	0	0 0 0 0 0 0 1 0 0	0	1:0 0 0 1 0 1 0 1	

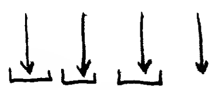
NOA WHEN Z_1^1 ILLUSTRATIVE EXAMPLE

OPERAND (LOADED IN D FROM MEMORY) 000000011

DATA (LEFT IN A FROM PREVIOUS INSTRUCTION) . . 000101010

OVERFLOW STATE (LEFT IN Z FROM PREVIOUS INSTRUCTION) . . 1

CONFIGURATION



DERIVATIVE INFORMATION FOR THE CITED EXAMPLE

ROMAN NUMERAL (SIGN QUARTER) I

SUBSCRIPTED ROMAN NUMERALS . . . . I_1 (=I)
(ACTIVE QUARTERS WHICH HAVE RN FOR SIGN QUARTER)

ASK PRESET 170

NOA – ASSUMING NO OVERFLOW OCCURRED IN PREVIOUS INSTRUCTION									
QK		AK	ASK	Y_1	D	Z_1	A_1	OPERATION	
14d	START	00d	X X X X X X X	X	X X X X X X X X X	0	0:0 0 1 0 1 0 1 0	CLEAR ASK	
21d	$E^1 \to D$	01d	0 0 0 0 0 0 0	0	0 0 0 0 0 0 0 0 0	0	0:0 0 1 0 1 0 1 0	PRESET ASK; LOAD D	
		02d	1 1 1 1 0 0 0	0	0 0 0 0 0 0 0 1 1	0	0:0 0 1 0 1 0 1 0		
		03d	1 1 1 1 0 0 0	0	0 0 0 0 0 0 0 1 1	0	0:0 0 0 1 0 1 0 1 0	COMPLEMENT D	
		04d	1 1 1 1 0 0 0	0	1 1 1 1 1 1 1 0 0	0	0:0 0 1 0 1 0 1 0	G_1	"NORMALIZE"
		04d	1 1 1 1 0 0 1	0	1 1 1 1 1 1 1 0 1	0	0:0 1 0 1 0 1 0 0	G_1	SHIFT LEFT IN A
		04d	1 1 1 1 0 1 0	0	1 1 1 1 1 1 1 1 0	0	0:1 0 1 0 1 0 0 0	$\overline{G}_1$	COUNT IN D
		00d	1 1 1 1 0 1 1	0	0 0 0 0 0 0 0 0 1	0	0:1 0 1 0 1 0 0 0		

NOA WHEN Z_1^0 ILLUSTRATIVE EXAMPLE

(SAME DATA AND INSTRUCTION SPECIFICATION

AS ABOVE, EXCEPT $Z_1 = 0$)

NORMALIZE A, AB

PK	24	α	⌊100→ PK

	00	α	QI^{start} . . . . ⊃ ⌊start◇ FK, ⌊0→ PI_1
	01	α	
			SEE QKM TIMING
	09	α	
	10	α	⌊0→ E
	11	α	⌊13→ QK D —1→ E
		β	⌊P→ E
QK	13	α	$M \xrightarrow[a,p]{0,1} E$ ⌊0→ AK_{11-1}, ⌊1→ AK_0 $QKIRcf_{3-8}$ —j→ $AKIRcf_{3-8}$ $QKIR^{ext\ act_{4-1}}$ —j→ $AKIRcf_{2-4}$ $QKIR_{op_{6-1}}$ —j→ $AKIR_{op_{6-1}}$
		β	⌊P→ E
	14	α	⌊21→ QK ⌊0→ E (SE) ⌊1→ AEP ⌊0→ D ⌊start◇ AK
		β	⌊C→ E (SE)
	21	α	E —1→ D
	22	α	
	23	α	$M \xrightarrow{0,1} E$ ⌊0→ EB ⌊31→ QK
	31	α	

OP Class Decoder Lines Up: $QKIR^{AK}$, $QKIR^{AESK}$, $QKIR^{load}$, $QKIR^{ld}$ & $QKIR^{D}$

OPERATE (ARITHMETIC ELEMENT: $N^0_{2.8} \cdot N^1_{2.7}$) OPR^{AE} 04

PK	24	α		
	25	α	QB^1 + AEB . . . . ⊃ AK^1_0 . . . . ⊃ " . . . . ⊃ " . . . . ⊃ (NOTE: AEB = AK^0_0)	$\overline{PK'}$+1 —+→ PK ⌊0→ AK_{11-1}, ⌊1→ AK_0 $N_{2.6-2.1}$ —j→ $AKIR_{op_{6-1}}$ $N_{1.9-1.4}$ —j→ $AKIRcf_{3-4}$
	26	α		⌊31→ QK ⌊1→ AEP ⌊start◇ AK
	31	α	$PI^{ch\ seq}$ . . . . ⊃	⌊1→ PI_3

OP Class Decoder Lines Up: $PKIR^{def}$, $PKIR^{dis}$, $PKIR^{AE}$

			4TH QUARTER	3RD QUARTER	2ND QUARTER	1ST QUARTER
	00	α				⌊start◇ AK . . . ⊃ $\overline{AK'}$+1→ AK ⌊start◇ AK . . . ⊃ ⌊0→ ASK
	01	α				⌊preset→ ASK
AK	02	α	$(Z^1_4 \cdot IV)$ . . . ⊃ ⌊C→ $A_{4.9}$ $(Z^1_4 \cdot a^1_4)$ . . . ⊃ ⌊SHR→ A_4 (") · $AKIR^{nab}$ ⊃ ⌊SHR→ B_4 $(Z^1_4 \cdot IV)$ . . . ⊃ $\overline{D'_4}$+1→ D_4	$(Z^1_3 \cdot III)$ . . . ⊃ ⌊C→ $A_{3.9}$ $(Z^1_3 \cdot III) + (Z^1_4 \cdot IV_3)$ . . . ⊃ ⌊SHR→ A_3 (") + (") · $AKIR^{nab}$ ⊃ ⌊SHR→ B_3 $(Z^1_3 \cdot III)$ . . . ⊃ $\overline{D'_3}$+1→ D_3	$(Z^1_2 \cdot II)$ . . . ⊃ ⌊C→ $A_{2.9}$ $(Z^1_2 \cdot II_2) + (Z^1_4 \cdot IV_2)$ . . . ⊃ ⌊SHR→ A_2 (") + (") · $AKIR^{nab}$ ⊃ ⌊SHR→ B_2 $(Z^1_2 \cdot II)$ . . . ⊃ $\overline{D'_2}$+1→ D_2	$(Z^1_1 \cdot I)$ . . . ⊃ ⌊C→ $A_{1.9}$ $(Z^1_1 \cdot I) + (Z^1_4 \cdot IV_1) + (Z^1_2 \cdot II_1)$ . . . ⊃ ⌊SHR→ A_1 (") + (") + (") · $AKIR^{nab}$ ⊃ ⌊SHR→ B_1 $(Z^1_1 \cdot I)$ . . . ⊃ $\overline{D'_1}$+1→ D_1
	03	α	IV . . . ⊃ ⌊0→ Z_4 IV . . . ⊃ ⌊C→ D_3	III . . . ⊃ ⌊0→ Z_3 III . . . ⊃ ⌊C→ D_3	II . . . ⊃ ⌊0→ Z_2 II . . . ⊃ ⌊C→ D_2	I . . . ⊃ ⌊0→ Z_1 I . . . ⊃ ⌊C→ D_1
	04	α	$(A_{4.9}=A_{4.8} \cdot a^1_4)$ . . ⊃ ⌊SHL→ A_4 (") · $AKIR^{nab}$ ⊃ ⌊SHL→ B_4 $(A_{4.9}=A_{4.8} \cdot IV)$ . . ⊃ $\overline{D'_4}$+1→ D_4 [SAME AS 4TH QUARTER] ⊃ ⌊C→ D_4	$(A_{3.9}=A_{3.8} \cdot III) + (A_{4.9}=A_{4.8} \cdot IV_3)$ . . ⊃ ⌊SHL→ A_3 (") + (") · $AKIR^{nab}$ ⊃ ⌊SHL→ B_3 $(A_{3.9}=A_{3.8} \cdot III)$ . . ⊃ $\overline{D'_3}$+1→ D_3 [SAME AS 4TH QUARTER] ⊃ ⌊C→ D_3	$(A_{2.9}=A_{2.8} \cdot II_2) + (A_{4.9}=A_{4.8} \cdot IV_2)$ . . ⊃ ⌊SHL→ A_2 (") + (") · $AKIR^{nab}$ ⊃ ⌊SHL→ B_2 $(A_{2.9}=A_{2.8} \cdot II)$ . . ⊃ $\overline{D'_2}$+1→ D_2 [SAME AS 4TH QUARTER] ⊃ ⌊C→ D_2	$\overline{ASK'}$+1→ ASK $\overline{AK'}$+1→ AK $\{[A_{4.9}\neq A_{4.8} \cdot (A_{3.9}\neq A_{3.8}+\overline{III})] + [A_{3.9}\neq A_{3.8} \cdot \overline{IV}] + [\overline{IV} \cdot \overline{III}]\} \cdot \{[A_{2.9}\neq A_{2.8} \cdot (A_{1.9}\neq A_{1.8}+\overline{I})] + [A_{1.9}\neq A_{1.8} \cdot \overline{II}] + [\overline{II} \cdot \overline{I}]\} + (ASK^0_7 \cdot ASK^0_6)$ ⊃ ⌊0→ AK_{11-1}, ⌊1→ AK_0 $\{[(\overline{A^0_4} \cdot \overline{A^1_4}) + \overline{IV}] \cdot [(\overline{A^0_3} \cdot \overline{A^1_3}) + \overline{III}] \cdot [(\overline{A^0_2} \cdot \overline{A^1_2}) + \overline{II}] \cdot [(\overline{A^0_1} \cdot \overline{A^1_1}) + \overline{I}]\} + \{ASK^1_7 \cdot ASK^1_6 \cdot ASK^1_5 \cdot ASK^1_4 \cdot ASK^0_3 \cdot ASK^1_2\}$ ⊃ ⌊0→ AEP $(A_{1.9}=A_{1.8} \cdot I) + (A_{4.9}=A_{4.8} \cdot IV_1) + (A_{2.9}=A_{2.8} \cdot II_1)$ . . ⊃ ⌊SHL→ A_1 (") + (") + (") · $AKIR^{nab}$ ⊃ ⌊SHL→ B_1 $(A_{1.9}=A_{1.8} \cdot I)$ . . ⊃ $\overline{D'_1}$+1→ D_1 $\{[A_{4.9}\neq A_{4.8} \cdot (A_{3.9}\neq A_{3.8}+\overline{III})] + [A_{3.9}\neq A_{3.8} \cdot \overline{IV}] + [\overline{IV} \cdot \overline{III}]\} \cdot$ $\{[A_{2.9}\neq A_{2.8} \cdot (A_{1.9}\neq A_{1.8}+\overline{I})] + [A_{1.9}\neq A_{1.8} \cdot \overline{II}] + [\overline{II} \cdot \overline{I}]\} +$ $(ASK^0_7 \cdot ASK^0_6)$ . . . ⊃ ⌊C→ D_1

CAN 6-23-61

DISTINGUISH A (FROM MEMORY)

OP CODE DESCRIPTION. DSA "partially-adds" the content of the selected Memory Element register to the content of the A register. The partial sum is left in A and the carries are left in C. Logically the OP code is defined as

$$\text{DSA} = \begin{cases} A \oplus Y_{CF} \longrightarrow A \\ C + A \cdot Y_{CF} \longrightarrow C \end{cases}$$

SPECIAL FEATURES. Partial addition is performed by a "pad" pulse. No coupling units are used.

DETAILS. The clearing and presetting of ASK are unnecessary operations, since ASK levels are not used in the instruction. Note that C is not cleared before the partial addition is performed. This results in the "carries" accumulating in C. Thus, in the example the partial sum of $A_{1.2}$ and $D_{1.2}$ produces a carry, but $C_{1.2}$ already contains a ONE, therefore $C_{1.2}$ is not affected. On the other hand, the partial addition of $A_{1.6}$ and $D_{1.6}$ produces a carry which appears in $C_{1.6}$. ($C_{1.6}$ was previously ZERO.)

Note that this instruction is simply an abbreviated ADD instruction.

See also ADD (67) and SUB (77) discussion.

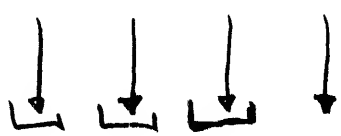
DSA						
QY		AK	D_1	C_1	A_1	OPERATION
14d	START	00d	X X X : X X X : X X X	0 1 0 : 0 0 0 : 0 1 0	0 0 0 : 1 0 1 : 0 1 0	
21d	$E \xrightarrow{1} D$	01d	0 0 0 : 0 0 0 : 0 0 0	0 1 0 : 0 0 0 : 0 1 0	0 0 0 : 1 0 1 : 0 1 0	LOAD D
		02d	0 0 0 : 1 0 0 : 0 1 1	0 1 0 : 0 0 0 : 0 1 0	0 0 0 : 1 0 1 : 0 1 0	
		03 α	0 0 0 : 1 0 0 : 0 1 1	0 1 0 : 0 0 0 : 0 1 0	0 0 0 : 1 0 1 : 0 1 0	PARTIAL ADDITION
		03 β	0 0 0 : 1 0 0 : 0 1 1	0 1 0 : 0 0 0 : 0 1 0	0 0 0 : 1 0 1 : 0 1 0	
		00d	0 0 0 : 1 0 0 : 0 1 1	0 1 0 : 1 0 0 : 0 1 0	0 0 0 : 0 0 1 : 0 0 1	

DSA ILLUSTRATIVE EXAMPLE

OPERAND (LOADED IN D FROM MEMORY) . . . . 0 0 0 : 1 0 0 : 0 1 1

DATA (LEFT IN A FROM PREVIOUS INST.) . . . . . 0 0 0 : 1 0 1 : 0 1 0
(" " C " " ") . . . . . 0 1 0 : 0 0 0 : 0 1 0

CONFIGURATION



DSA (65)

DISTINGUISH A (FROM MEMORY)

PK	24	α	$\underrightarrow{100}$ PK

	00	α	QI^{start} · · · · · · · · · ⊃ $\underrightarrow{start}$◇ FK , $\underrightarrow{L0}$ PI_1
	01	α	
			SEE QKM TIMING
	09	α	
	10	α	$\underrightarrow{L0}$ E
	11	α	$\underrightarrow{L13}$ QK $D \xrightarrow{1} E$
		β	$\underrightarrow{LP}$ E
QK	13	α	$M \xrightarrow[a,p]{0,1} E$ $\underrightarrow{L0}$ AK_{11-1} , $\underrightarrow{L1}$ AK_0 $QKIR_{cf_{9-8}} \xrightarrow{j} AKIR_{cf_{9-8}}$ $QKIR^{extract}_{4-1} \xrightarrow{j} AKIR_{cf_{7-4}}$ $QKIR_{op_{6-1}} \xrightarrow{j} AKIR_{op_{6-1}}$
		β	$\underrightarrow{LP}$ E
	14	α	$\underrightarrow{L21}$ QK $\underrightarrow{L0}_{SE}$ E $\underrightarrow{L1}$ AEP $\underrightarrow{L0}$ D $\underrightarrow{start}$◇ AK
		β	$\underrightarrow{LC}_{SE}$ E
	21	α	$E \xrightarrow{1} D$
	22	α	
	23	α	$M \xrightarrow{0,1} E$ $\underrightarrow{L0}$ EB $\underrightarrow{L31}$ QB
	31	α	

OP Class Decoder Lines Up: $QKIR^{AK}$, $QKIR^{AESK}$, $QKIR^{load}$, $QKIR^{ld}$ + $QKIR^{D}$

OPERATE (ARITHMETIC ELEMENT: $N^0_{2\cdot 8} \cdot N^1_{2\cdot 7}$) OPR^{AE} 04

	24	α	
PK	25	α	QB' + AEB · · · · · · · ⊃ $\overline{PK'} + 1 \not\rightarrow$ PK AK'_0 · · · · · · · · · · ⊃ $\underrightarrow{L0}$ AK_{11-1} , $\underrightarrow{L1}$ AK_0 " · · · · · · · · · · ⊃ $N_{2.6-2.1} \xrightarrow{j} AKIR_{op_{6-1}}$ " · · · · · · · · · · ⊃ $N_{1.9-1.4} \xrightarrow{j} AKIR_{cf_{9-4}}$ (Note: AEB = AK^0_0)
	26	α	$\underrightarrow{L31}$ QK $\underrightarrow{L1}$ AEP $\underrightarrow{start}$◇ AK
	31	α	$PI^{ch\ seq}$ · · · · · · · · · ⊃ $\underrightarrow{L1}$ PI_3

OP Class Decoder Lines Up: $PKIR^{def}$, $PKIR^{dis}$, $PKIR^{AE}$

AK	00	α				$\underrightarrow{start}$◇ AK · · · · ⊃ $\overline{AK'} + 1 \longrightarrow$ AK $\underrightarrow{start}$◇ AK · · · · ⊃ $\underrightarrow{L0}$ ASK
	01	α				$\underrightarrow{preset}$ ASK $\underrightarrow{L0}$ AEP
	02	α				
	03	α				$\underrightarrow{L0}$ AK_{11-1} , $\underrightarrow{L1}$ AK_0 $\overline{AK'} + 1 \not\rightarrow$ AK
			4TH QUARTER	3RD QUARTER	2ND QUARTER	1ST QUARTER
		β	a'_4 · · · · ⊃ $\underrightarrow{pad}$ A_4, C_4	a'_3 · · · · · ⊃ $\underrightarrow{pad}$ A_3, C_3	a'_2 · · · · ⊃ $\underrightarrow{pad}$ A_2, C_2	a'_1 · · · · · ⊃ $\underrightarrow{pad}$ A_1, C_1

ADD (MEMORY TO A)

SUBTRACT (MEMORY FROM A)

OP CODE DESCRIPTION. The content of the selected Memory Element register is ADDed (SUBtracted) from the content of the A register. If an overflow occurs, it is indicated by the Z overflow flip-flops in the sign quarters of A.

SPECIAL FEATURES. Overflow logic is used in controlling the state of the Z flip-flop in the sign quarter(s). The addition (subtraction) is performed by "partial addition" and "carry"logic. These instructions use Carry Coupling Units.

DETAILS. The clearing and presetting of the ASK counter is an unnecessary operation, since ASK levels are not used in the instruction.

The active subwords of C are cleared preliminary to the partial addition operation. (This pulse does not occur during a DSA.)

If a subtraction is involved the active subwords of D are complemented. This is the only pulse where an explicit distinction is made between the ADDition and SUBtraction logic.

The content of A and D are partially added in $AK^{03\beta}$. The partial sum appears in A and the carries in C. (During a DSA, AK does not progress beyond this state.)

ADD
67

A complete carry is propagated through the active quarters of A. The complete sum appears in A after this operation.

The last step in the logic forces the sign of D to agree with the sign of the original operand as remembered by Y. (The only time they can differ is when a SUBtraction is executed.)

The logic controlling Z is complex and is described in Chapter 14. Note that in both the ADD and SUB examples, Z is cleared and set early in the AK instruction. If no overflow has occurred, Z is cleared by the reset Z logic in the last AK state. In the ADDition example no overflow occurs and Z is cleared. However, in the SUBtraction example an overflow does occur and Z is left set. The logic selects the Z flip-flop associated with the sign quarter in A, i.e., the quarters selected by the Roman numeral levels.

SUB
77

ADD

QK		AK	Y_1	D_1	C_1	Z_1	A_1	OPERATION
14_8	START	00_8	X	XXXXXXXXX	XXXXXXXXX	X	000101010	
21_8	$E \xrightarrow{1} D$	01_8	0	000000000	XXXXXXXXX	X	000101010	CLEAR C&Z LOAD D
		02_8	0	000100011	000000000	0	000101010	
		03 α	0	000100011	000000000	0	000101010	PARTIAL ADDITION
		03 β	0	000100011	000000000	1	000101010	
		04_8						
		05_8						
		06_8	0	000100011	000100010	1	000001001	
		07_8	0	000100011	000100010	1	000001001	
		08_8	0	000100011	000100010	1	000001001	CARRY
		09_8	0	000100011	000100010	1	001001101	SIGN & OVERFLOW FIXUP
		00_8	0	000100011	000100010	0	001001101	

ADD ILLUSTRATIVE EXAMPLE

OPERAND (LOADED IN D FROM MEMORY) 000100011

DATA (LEFT IN A FROM PREVIOUS INSTRUCTION). . 000101010

CONFIGURATION

DERIVATIVE INFORMATION FOR THE CITED EXAMPLE

ROMAN NUMERAL (SIGN QUARTER) I

SUBSCRIPTED ROMAN NUMERAL (SUBSCRIPT = ACTIVE QUARTER WITH RN FOR SIGN QUARTER) . . . . I_1 (= I)

SUB

QK		AK	Y_1	D_1	C_1	Z_1	A_1	OPERATION
14_8	START	00_8	X	XXXXXXXXX	XXXXXXXXX	X	000101010	
21_8	$E \xrightarrow{1} D$	01_8	0	000000000	XXXXXXXXX	X	000101010	CLEAR C&Z
		02_8	0	000100011	000000000	0	000101010	COMPLEMENT D
		03 α	0	111011100	000000000	0	000101010	PARTIAL ADDITION
		03 β	0	111011100	000000000	1	000101010	
		04_8						
		05_8						
		06_8	0	111011100	000001000	1	111110110	
		07_8	0	111011100	000001000	1	111110110	
		08_8	0	111011100	000001000	1	111110110	CARRY
		09_8	0	111011100	000001000	1	000000111	SIGN & OVERFLOW FIXUP
		00_8	0	000100011	000001000	1	000000111	

SUB ILLUSTRATIVE EXAMPLE

(SAME DATA AND SPECIFICATION AS ADD EXAMPLE ABOVE)

ADD, SUB (67, 77)

ADD (MEMORY TO A); SUBTRACT (MEMORY FROM A)

PK	24	α		$\xrightarrow{00} PK$

	00	α	QI^{start} . . . . $\supset$	$\xrightarrow{start}\diamond AK$, $\xrightarrow{0} PI_1$
	01	α		
	SEE QKM TIMING			
	09	α		
	10	α		$\xrightarrow{0} E$
	11	α		$\xrightarrow{13} QK$ $D \xrightarrow{1} E$
		β		$\xrightarrow{\bar{P}} E$
QK	13	α		$M \xrightarrow[a,p]{0,1} E$ $\xrightarrow{0} AK_{11-1}$, $\xrightarrow{1} AK_0$ $QKIR_{cf_{9-8}} \xrightarrow{j} AKIR_{cf_{9-8}}$ $QKIR^{extract}_{4-1} \xrightarrow{j} AKIR_{cf_{7-4}}$ $QKIR_{op_{6-1}} \xrightarrow{j} AKIR_{op_{6-1}}$
		β		$\xrightarrow{P} E$
	14	α		$\xrightarrow{21} QK$ $\xrightarrow[SE]{0} E$ $\xrightarrow{1} AEP$ $\xrightarrow{0} D$ $\xrightarrow{start}\diamond AK$
		β		$\xrightarrow[SE]{c} E$
	21	α		$E \xrightarrow{1} D$
	22	α		
	23	α		$M \xrightarrow{0,1} E$ $\xrightarrow{0} EB$ $\xrightarrow{31} QK$
	31	α		

OP Class Decoder Lines Up: $QKIR^{AK}$, $QKIR^{AESK}$, $QKIR^{load}$, $QKIR^{ld}$ & $QKIR^{D}$

OPERATE (ARITHMETIC ELEMENT: $N^{0}_{2.8} \cdot N^{1}_{2.7}$) OPR^{AE} 04

24	α		
25	α	QB^{1} + AEB . . . . $\supset$ AK^{1}_{0} . . . . $\supset$ " . . . . $\supset$ " . . . . $\supset$ (NOTE: $AEB = AK^{0}_{0}$)	$\overline{PK'}+1 \not\rightarrow PK$ $\xrightarrow{0} AK_{11-1}$, $\xrightarrow{1} AK_0$ $N_{2.6-2.1} \xrightarrow{j} AKIR_{op_{6-1}}$ $N_{1.9-1.4} \xrightarrow{j} AKIR_{cf_{9-4}}$
26	α		$\xrightarrow{31} QK$ $\xrightarrow{1} AEP$ $\xrightarrow{start}\diamond AK$
31	α	$PI^{ch\ seq}$. . . . . $\supset$	$\xrightarrow{1} PI_3$

OP Class Decoder Lines Up: $PKIR^{def}$, $PKIR^{dis}$, $PKIR^{AE}$

AK	00	α				$\xrightarrow{start}\diamond AK^{\cdot}$ . . . $\supset$ $\overline{AK}+1 \rightarrow AK$ $\xrightarrow{start}\diamond AK'$ . . . $\supset$ $\xrightarrow{0} ASK$
	01	α				$\xrightarrow{preset} ASK$ $\xrightarrow{0} AEP$
			4TH QUARTER	3RD QUARTER	2ND QUARTER	1ST QUARTER
			IV . . . . $\supset$ $\xrightarrow{0} Z_4$ a^{1}_{4} . . . . $\supset$ $\xrightarrow{0} C_4$	III . . . . $\supset$ $\xrightarrow{0} Z_3$ a^{1}_{3} . . . . $\supset$ $\xrightarrow{0} C_3$	II . . . . $\supset$ $\xrightarrow{0} Z_2$ a^{1}_{2} . . . . $\supset$ $\xrightarrow{0} C_2$	I . . . . $\supset$ $\xrightarrow{0} Z_1$ a^{1}_{1} . . . . $\supset$ $\xrightarrow{0} C_1$
	02	α	$AKIR^{sub} \cdot a^{1}_{4}$ . . . . . $\supset$ $\xrightarrow{c} D_4$	$AKIR^{sub} \cdot (III + IV_3)$ . . $\supset$ $\xrightarrow{c} D_3$	$AKIR^{sub} \cdot (II_2 + IV_2)$ . . $\supset$ $\xrightarrow{c} D_2$	$AKIR^{sub} \cdot (I + IV_1 + II_1)$ . . . $\supset$ $\xrightarrow{c} D_1$
	03	α	IV . . . . $\supset$ $\xrightarrow{pad} Z_4$	III . . . . $\supset$ $\xrightarrow{pad} Z_3$	II . . . . $\supset$ $\xrightarrow{pad} Z_2$	$\overline{AK}+1 \not\rightarrow AK$ $(AK^{0}_{\alpha.8} \cdot \bar{f}_4)$ . . . . $\supset$ $\xrightarrow{05} AK$ $(AK^{0}_{\alpha.8} \cdot f_4)$ . . . . $\supset$ $\xrightarrow{06} AK$ I . . . . $\supset$ $\xrightarrow{pad} Z_1$
		β	a^{1}_{4} . . . . $\supset$ $\xrightarrow{pad} A_4, C_4$	a^{1}_{3} . . . . $\supset$ $\xrightarrow{pad} A_3, C_3$	a^{1}_{2} . . . . $\supset$ $\xrightarrow{pad} A_2, C_2$	a^{1}_{1} . . . . $\supset$ $\xrightarrow{pad} A_1, C_1$
	05	α				
	06	α				
	07	α				
	08	α	$\bar{A}^{1}_{4}+(\bar{A}^{1}_{3} \cdot \bar{f}_4)+[\bar{A}^{1}_{2} \cdot (f_1+f_3)]+(\bar{A}^{1}_{1} \cdot f_1) \supset \xrightarrow{CRY} A_4$	$(\bar{A}^{1}_{4} \cdot \bar{f}_4)+\bar{A}^{1}_{3}+[\bar{A}^{1}_{2} \cdot (f_1+f_3)]+(\bar{A}^{1}_{1} \cdot f_1) \supset \xrightarrow{CRY} A_3$	$[(\bar{A}^{1}_{4}+\bar{A}^{1}_{3}) \cdot (f_1+f_3)]+\bar{A}^{1}_{2}+[\bar{A}^{1}_{1} \cdot (f_1+f_2)] \supset \xrightarrow{CRY} A_2$	$[(\bar{A}^{1}_{4}+\bar{A}^{1}_{3}) \cdot f_1]+[\bar{A}^{1}_{2} \cdot (f_1+f_2)]+\bar{A}^{1}_{1}$ . . $\supset \xrightarrow{CRY} A_1$
	09	α	IV . . . . $\supset$ $\xrightarrow{reset} Z_4$ $(Y_4 \neq D_{4.9}) \cdot a^{1}_{4}$ . . . . $\supset$ $\xrightarrow{c} D_4$	III . . . . $\supset$ $\xrightarrow{reset} Z_3$ $[(Y_3 \neq D_{3.9}) \cdot III]+[(Y_4 \neq D_{4.9})+IV_3] \supset \xrightarrow{c} D_3$	II . . . . $\supset$ $\xrightarrow{reset} Z_2$ $[(Y_2 \neq D_{2.9}) \cdot II_2]+[(Y_4 \neq D_{4.9}) \cdot IV_2] \supset \xrightarrow{c} D_2$	$\overline{AK}+1 \not\rightarrow AK$ $\xrightarrow{0} AK_{11-1}$ $\xrightarrow{1} AK_0$ I . . . . $\supset$ $\xrightarrow{reset} Z_1$ $[(Y_1 \neq D_{1.9}) \cdot I]+[(Y_4 \neq D_{4.9}) \cdot IV_1]+[(Y_2 \neq D_{2.9}) \cdot II_1] \supset \xrightarrow{c} D_1$

CAN 6-26-61

TALLY (ONES IN MEMORY)

OP CODE DESCRIPTION. TLY examines the content of the selected Memory Element register for ONES. The number of ONES appearing in active subwords is added to the content of the corresponding sign quarters of the D register. Except for the effect on D, the final result is as if a LDA instruction were performed.

SPECIAL FEATURES. The operand is loaded into A instead of into D as is normally done in AK type instructions. This instruction uses the Shift Coupling Unit.

DETAILS. ASK determines the number of shifts that will occur in A. The number of shifts equals the length of the subword. ASK is preset to a value determined by the fracture specification of the instruction. In the example, ASK is preset to 170, since an F_4 (9,9,9,9) fracture is specified. For each shift in A, ONE is added to the contents of ASK. The final value of ASK is always 001. Before each shift the state of the sign bit of A is examined. If a ONE is sampled, the corresponding (sign) quarter of the D register is indexed. (D acts as a counter.)

In the example, A_1 contains three (3) ONES. Note that D contains 006 at the end of the instruction and 003 at the beginning of the instruction, i.e., the accumulated count in D is 003.

The subwords in A are rotated, as in a CYcle instruction, so that the content of A at the end of the instruction is exactly the same as after the operand was originally loaded in A.

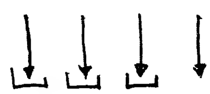
TLY						
QK		AK	ASK	D_1	A_1	OPERATION
14d	START	00d	X X X X X X X	0 0 0 0 0 0 0 1 1	X X X X X X X X X	CLEAR ASK
21d	E→A	01d	0 0 0 0 0 0 0	0 0 0 0 0 0 0 1 1	0 0 0 0 0 0 0 0 0	PRESET ASK; LOAD A
		02d	1 1 1 1 0 0 0	0 0 0 0 0 0 0 1 1	0 0 0 1 0 1 0 1 0	"TALLY"
		02d	1 1 1 1 0 0 1	0 0 0 0 0 0 0 1 1	0 0 0 0 1 0 1 0 1	
		02d	1 1 1 1 0 1 0	0 0 0 0 0 0 0 1 1	1 0 0 0 0 1 0 1 0	SHIFT THE CON-
		02d	1 1 1 1 0 1 1	0 0 0 0 0 0 1 0 0	0 1 0 0 0 0 1 0 1	TENTS OF A AND
		02d	1 1 1 1 1 0 0	0 0 0 0 0 0 1 0 0	1 0 1 0 0 0 0 1 0	SAMPLE $A_{i.q}$ for
		02d	1 1 1 1 1 0 1	0 0 0 0 0 0 1 0 1	0 1 0 1 0 0 0 0 1	ONES. COUNT
		02d	1 1 1 1 1 1 0	0 0 0 0 0 0 1 0 1	1 0 1 0 1 0 0 0 0	THE ONES SAM-
		02d	1 1 1 1 1 1 1	0 0 0 0 0 0 1 1 0	0 1 0 1 0 1 0 0 0	PLED IN D
		02d	0 0 0 0 0 0 0	0 0 0 0 0 0 1 1 0	0 0 1 0 1 0 1 0 0	
		00d	0 0 0 0 0 0 1	0 0 0 0 0 0 1 1 0	0 0 0 1 0 1 0 1 0	

TLY ILLUSTRATIVE EXAMPLE

OPERAND (LOADED IN A FROM MEMORY) 000 101 010

DATA (LEFT IN D FROM PREVIOUS INSTRUCTION) . . . 000 000 011

CONFIGURATION



DERIVATIVE INFORMATION FOR THE CITED EXAMPLE:

ROMAN NUMERAL (SIGN QUARTER) I

ASK PRESET 170

TALLY (ONES IN MEMORY)

PK	24	α		100 → PK

	00	α	QI^{start} . . . . ⊃	start ◇ FK, L0 → PI_1
	01	α		
	SEE QKM TIMING			
	09	α		
	10	α		L0 → E
	11	α		L13 → QK A —1→ E
	11	β		LP̄ → E
QK	13	α		M $\xrightarrow[a,P]{0,1}$ E L0 → AK_{11-1}, L1 → AK_0 $QKIR_{cf_{9-8}}$ —j→ $AKIR_{cf_{9-8}}$ $QKIR^{extract_{4-1}}$ —j→ $AKIR_{cf_{7-4}}$ $QKIR_{op_{6-1}}$ —j→ $AKIR_{op_{6-1}}$
	13	β		LP̄ → E
	14	α		L21 → QK $L0_{SE}$ → E L1 → AEP L0 → A start ◇ AK
	14	β		Lc_{SE} → E
	21	α		E —1→ A
	22	α		
	23	α		M —0,1→ E L0 → EB L31 → QK
	31	α		

OP Class Decoder Lines Up: $QKIR^{AK}$, $QKIR^{AESK}$, $QKIR^{load}$, $QKIR^{ld}$ & $QKIR^{A}$

OPERATE (ARITHMETIC ELEMENT: $N^0_{2.8} \cdot N^1_{2.7}$) OPR^{AE} 04

	24	α		
PK	25	α	QB′ + AEB . . . . . . ⊃ AK^1_0 . . . . ⊃ " . . . . ⊃ " . . . . ⊃ (NOTE: AEB = AK^0_0)	$\overline{PK}'$ + 1 —/→ PK L0 → AK_{11-1}, L1 → AK_0 $N_{2.6-2.1}$ —j→ $AKIR_{op_{6-1}}$ $N_{1.9-1.4}$ —j→ $AKIR_{cf_{9-4}}$
	26	α		L31 → QK L1 → AEP start ◇ AK
	31	α	$PI^{ch\ seq}$ . . . . ⊃	L1 → PI_3

OP Class Decoder Lines Up: $PKIR^{def}$, $PKIR^{dis}$, $PKIR^{AE}$

AK	00	α	start ◇ AK . . . . . ⊃ start ◇ AK . . . . . ⊃	$\overline{AK}'$ + 1 —→ AK L0 → ASK
	01	α		preset → ASK
	02	α	$ASK^0_7 \cdot ASK^0_6$ . . . . ⊃ $ASK^1_7 \cdot ASK^1_6 \cdot ASK^1_5 \cdot ASK^1_4 \cdot ASK^1_3 \cdot ASK^1_2$ . . . . ⊃	$\overline{ASK}'$ + 1 —→ ASK $\overline{AK}'$ + 1 —/→ AK L0 → AK_{11-1}, L1 → AK_0 L0 → AEP

4TH QUARTER	3RD QUARTER	2ND QUARTER	1ST QUARTER
a^1_4 . . . . . ⊃ SHR → A_4 IV · $A^1_{4.9}$ . . . ⊃ $\overline{D}'_4$ + 1 → D_4	a^1_3 . . . . . ⊃ SHR → A_3 III · $A^1_{3.9}$ . . . ⊃ $\overline{D}'_3$ + 1 → D_3	a^1_2 . . . . . ⊃ SHR → A_2 II · $A^1_{2.9}$ . . . ⊃ $\overline{D}'_2$ + 1 → D_2	a^1_1 . . . . . ⊃ SHR → A_1 I · $A^1_{1.9}$ . . . ⊃ $\overline{D}'_1$ + 1 → D_1

OP CODE DESCRIPTION. DIV divides the content of AB by the content of the selected Memory Element register. The quotient appears in A and the remainder in B. If an overflow occurs the Z flip-flops are set. With the exception of the possible generating of remainders and/or overflows the instruction is the inverse of MULtiply.

SPECIAL FEATURES. The ASK counter is used to count the number of carry (CRY), partial-add (PAD) loop iterations. In this instruction, the Z and Y flip-flops are used in the sign control logic although the Z flip-flop is also used in the DIV overflow control logic. DIV uses the Carry Coupling Units and the Shift Coupling Units.

DETAILS. The configured operand (divisor) is loaded into D at $QK^{21\alpha}$. The QK execution logic is identical to that of a LDD, except for the pulses indicated on the DIV time chart. $QK^{14\alpha}$ starts the AK counter which controls the division logic.

The pulses clearing and presetting the ASK counter occur at $AK^{00\alpha}$ and $AK^{01\alpha}$, respectively. In the example ASK is preset to 170. This value is determined by the length of the longest subword specified by the configuration.

The sign of the dividend is copied into Z and the content of AB is made negative at $AK^{01\alpha}$. The C register is also cleared as a preliminary to storing the partial carries in the succeeding steps. If A and D have the same sign then the content of D is made negative at $AK^{02\alpha}$. The first pad pulse is fired off in $AK^{02\beta}$.

The CRY-PAD loop is now entered. Each time the loop is traversed a count ASK pulse occurs.

Certain characteristics of the CRY-PAD loop should be pointed out. (The carry and partial add logic are fully explained in Chapter 14. An end-around carry does not occur in DIV; instead, the content of the sign bit in D is carried into the right end of the carry circuit of each subword. Note that a complete carry occurs in each traverse of the loop. Before the partial addition occurs, the sign of D is always made the complement of that of A. At the end of each CRY-PAD loop the content of AB is shifted (rotated) one bit to the left in each subword. The bits shifted into the right end of each subword in B generate the quotient. Note that this shift is made after the decision to traverse the CRY-PAD loop again. During the last loop ($ASK_7^1 \cdot ASK_1^0$ at AK^{09}) the content of B is shifted to the left, but <u>not</u> the content of A. Whether or not the pad pulse is fired off in this last loop is conditioned by the sign of the subwords in A. This last loop generates the correct remainder in A.

On the last traverse of the loop ASK is indexed so that $ASK_7^0 \cdot ASK_2^1 \cdot ASK_1^0$ is true, and AK jumps to $AK^{10\alpha}$.

DIV
75

$AK^{10\alpha}$ interchanges the content of A and B, placing the quotient in A and the remainder in B.

$AK^{11\alpha}$ takes care of the sign and overflow conditions. If the quotient is negative at $AK^{11\alpha}$, then an overflow occurred during the division process. Note that if an overflow occurs and the dividend <u>is less</u> than twice as large as the divisor, then the overflow can be shifted into A by a SCale or NOrmalize instruction and the correct quotient obtained.

DIVIDE ILLUSTRATIVE EXAMPLE

QK		AK	ASK	Y_1	D_1	C_1	Z_1	A_1	B_1	OPERATION	
19d	START	00d	xxxxxxxx	x	xxxxxxxxx	xxxxxxxxx	x	0101011100	0000001100	① Operand (Dividend) Loaded into D	
21d	E→D	01d	00000000	0	000000000	xxxxxxxxx	x	0101011100	0000001100	② ASK Preset	
		09 α	11111000	1	100111101	000000000	0	1010100011	1111110011	③ AB Made Negative ④ Sign of D & AB Made	
		09 β	11111000	1	011000010	000000000	0	1010100011	1111110011	PAD ⑤ Different PartialAdd Occurs	
		06d	11111000	1	011000010	001000010	0	1101010001	1111110011	CRY-SHL-PAD Loop	
		07d	11111000	1	011000010	001000010	0	1101010001	1111110011		
		08d	11111001	1	011000010	001000010	0	1101010001	1111110011	CRY	
		09 α	11111001	1	011000010	000000000	0	0000101101	1111110011	SHL	1
		09 β	11111001	1	100111101	000000000	0	0001010111	1111100110	PAD	
		06d	11111001	1	100111101	000101001	0	1000101110	1111100110		
		07d	11111001	1	100111101	000101001	0	1001010110	1111100110		
		08d	11111010	1	100111101	000101001	0	1001010110	1111100110	CRY	
		09 α	11111010	1	100111101	000000000	0	1011010000	1111100110	SHL	2
		09 β	11111010	1	011000010	000000000	0	0111010011	1111001101	PAD	
		06d	11111010	1	011000010	011000010	0	0000101001	1110011101		
		07d	11111010	1	011000010	011000010	0	0000101001	1110011101		
		08d	11111011	1	011000010	011000010	0	0000101001	1110011101	CRY	
		09 α	11111011	1	011000010	000000000	0	1101011101	1110011101	SHL	3
		09 β	11111011	1	011000010	000001000	0	1001010111	1100110111	PAD	
		06d	11111011	1	011000010	000000010	0	1111010001	1100110111		
		07d	11111011	1	011000010	000000010	0	1111101001	1100110111		
		08d	11111100	1	011000010	000000010	0	1111101001	1100110111	CRY	
		09 α	11111100	1	011000010	000000000	0	1111101101	1100110111	SHL	4
		09 β	11111100	1	011000010	000000000	0	1111011011	1001101111	PAD	
		06d	11111100	1	011000010	011000010	0	1001011001	1001101111		
		07d	11111100	1	011000010	011000010	0	1001011001	1001101111		
		08d	11111101	1	011000010	011000010	0	1001011001	1001101111	CRY	
		09 α	11111101	1	011000010	000000000	0	0101011101	1001101111	SHL	5
		09 β	11111101	1	100111101	000000000	0	1001110111	0011011110	PAD	
		06d	11111101	1	100111101	100111001	0	0000000110	0011011110		
		07d	11111101	1	100111101	100111001	0	0000000110	0011011110		
		08d	11111110	1	100111101	100111001	0	0000000110	0011011110	CRY	
		09 α	11111110	1	100111101	000000000	0	0011111001	0011011110	SHL	6
		09 β	11111110	1	100111101	000000000	0	0111100010	0110111100	PAD	
		06d	11111110	1	100111101	000110000	0	1110001111	0110111100		
		07d	11111110	1	100111101	000110000	0	1110011111	0110111100		
		08d	11111111	1	100111101	000110000	0	1110011111	0110111100	CRY	
		09 α	11111111	1	100111101	000000000	0	0001100000	0110111100	SHL	7
		09 β	11111111	1	100111101	000000000	0	0010000000	1101111000	PAD	
		06d	11111111	1	100111101	000100000	0	1101111101	1101111000		
		07d	11111111	1	100111101	000100000	0	1010111101	1101111000		
		08d	00000000	1	100111101	000100000	0	1010111101	1101111000	CRY	
		09 α	00000000	1	100111101	000000000	0	1101011110	1101111000	SHL	8
		09 β	00000000	1	011000010	000000000	0	1001111101	1011101001	PAD	
		06d	00000000	1	011000010	000100000	0	1111111111	1011100001	SHL does not occur in A	
		07d	00000000	1	011000010	000100000	0	1111111111	1011100001	PAD Pulse is conditional	
		08d	00000001	1	011000010	000000000	0	1111111111	1011100001	CRY	on sign of A.
		09 α	01000001	1	011000010	000000000	0	1111111111	1011100001	SHL	9
		09 β	01000001	1	011000010	000000000	0	1111111111	0111000011		
		06d	00000001	1	011000010	000000000	0	1111111111	0111000011		
		07d	00000001	1	011000010	000000000	0	1111111111	0111001011		
		08d	00000010	1	011000010	000000000	0	1111111111	0111001011	CRY	
		10d	00000010	1	011000010	000000000	0	1111111111	0111000011	A and B interchanged.	
		11d	00000010	1	011000010	000000000	0	0111000011	1111111111	Sign and overflow fix up	
		00d	00000010	1	100111101	000000000	0	1000111100	0000000000	occur.	

DIVIDE ILLUSTRATIVE EXAMPLE

OPERAND (LOADED IN D FROM MEMORY) 100111101

DIVISOR (LEFT IN AB FROM PREVIOUS INSTRUCTION) 0101011100 0000001100

FRACTURE

DIVIDE (75)

PK	24	α	$\lfloor 00 \to PK$

	00	α	QI^{start} . . . . ⊃ $\lfloor start \diamond FK$, $\lfloor 0 \to PI_1$
	01	α	
			SEE QKM TIMING
	09	α	
	10	α	$\lfloor 0 \to E$
	11	α	$\lfloor 13 \to QK$ $D \xrightarrow{1} E$
		β	$\lfloor \bar{P} \to E$
QK	13	α	$M \xrightarrow[a,P]{0,1} E$ $\lfloor 0 \to AK_{11-1}$, $\lfloor 1 \to AK_0$ $QKIR_{cf_{9-8}} \xrightarrow{j} AKIR_{cf_{9-8}}$ $QKIR^{extact_{4-1}} \xrightarrow{j} AKIR_{cf_{7-4}}$ $QKIR_{op_{6-1}} \xrightarrow{j} AKIR_{op_{6-1}}$
		β	$\lfloor P \to E$
	14	α	$\lfloor 21 \to QK$ $\lfloor 0_{SE} \to E$ $\lfloor 1 \to AEP$ $\lfloor 0 \to D$ $\lfloor start \diamond AK$
		β	$\lfloor C_{SE} \to E$
	21	α	$E \xrightarrow{1} D$
	22	α	
	23	α	$M \xrightarrow{0,1} E$ $\lfloor 0 \to EB$ $\lfloor 31 \to QK$
	31	α	

OP Class Decoder Lines Up: $QKIR^{AK}$, $QKIR^{AESK}$, $QKIR^{load}$, $QKIR^{ld}$ + $QKIR^{D}$

OPERATE (ARITHMETIC ELEMENT: $N^0_{2.8} \cdot N^1_{2.7}$) OPR^{AE} 04

PK	24	α	
	25	α	QB^1 + AEB . . . . ⊃ $\overline{PK'} + 1 \xrightarrow{j} PK$ AK^1_0 . . . . ⊃ $\lfloor 0 \to AK_{11-1}$, $\lfloor 1 \to AK_0$ " . . . . ⊃ $N_{2.6-2.1} \xrightarrow{j} AKIR_{op_{6-1}}$ " . . . . ⊃ $N_{1.9-1.4} \xrightarrow{j} AKIR_{cf_{9-4}}$ (NOTE: AEB = AK^0_0)
	26	α	$\lfloor 31 \to PK$ $\lfloor 1 \to AEP$ $\lfloor start \diamond AK$
	31	α	$PI^{ch\ seq}$ . . . . ⊃ $\lfloor 1 \to PI_3$

OP Class Decoder Lines Up: $PKIR^{def}$, $PKIR^{dis}$, $PKIR^{AE}$

			4TH QUARTER	3RD QUARTER	2ND QUARTER	1ST QUARTER
AK	00	α				$\lfloor start \diamond AK$ . . ⊃ $\overline{AK'}+1 \to AK$ $\lfloor start \diamond AK$ . . ⊃ $\lfloor 0 \to ASK$
	01	α	$(A^0_{4.9} \cdot a^1_4)$ . . ⊃ $\lfloor C \to A_4, B_4$ IV . . . . . ⊃ $\lfloor A_{4.9}\ sign \to Z_4$ a^1_4 . . . . . ⊃ $\lfloor C \to C_4$	$(A^0_{3.9} \cdot III) + (A^0_{4.9} \cdot IV_3)$. ⊃ $\lfloor C \to A_3, B_3$ III . . . . ⊃ $\lfloor A_{3.9}\ sign \to Z_3$ a^1_3 . . . . ⊃ $\lfloor C \to C_3$	$(A^0_{2.9} \cdot II_2) + (A^0_{4.9} \cdot IV_2)$ . . . ⊃ $\lfloor C \to A_2, B_2$ II . . . . ⊃ $\lfloor A_{2.9}\ sign \to Z_2$ a^1_2 . . . . ⊃ $\lfloor C \to C_2$	$\lfloor preset \to ASK$ $(A^0_{1.9} \cdot I) + (A^0_{4.9} \cdot IV_1) + (A^0_{2.9} \cdot II_1)$ . . ⊃ $\lfloor C \to A_1, B_1$ I . . . . ⊃ $\lfloor A_{1.9}\ sign \to Z_1$ a^1_1 . . . . ⊃ $\lfloor C \to C_1$
	02	α	$(A_{4.9} \equiv D_{4.9}) \cdot a^1_4$. ⊃ $\lfloor C \to D_4$	$[(A_{3.9} \equiv D_{3.9}) \cdot III] + [(A_{4.9} \equiv D_{4.9}) \cdot IV_3]$ ⊃ $\lfloor C \to D_3$	$[(A_{2.9} \equiv D_{2.9}) \cdot II_2] + [(A_{4.9} \equiv D_{4.9}) \cdot IV_2]$ ⊃ $\lfloor C \to D_2$	$\overline{AK'}+1 \xrightarrow{j} AK$ $\bar{f}_4$ . . . ⊃ $\lfloor 05 \to AK$ f_4 . . . ⊃ $\lfloor 06 \to AK$ $[(A_{1.9} \equiv D_{1.9}) \cdot I] + [(A_{4.9} \equiv D_{4.9}) \cdot IV_1] + [(A_{2.9} \equiv D_{2.9}) \cdot II_1]$ ⊃ $\lfloor C \to D_1$
		β	a^1_4 . . . . ⊃ $\lfloor C \to A_4, C_4$	a^1_3 . . . . ⊃ $\lfloor C \to A_3, C_3$	a^1_2 . . . . ⊃ $\lfloor C \to A_2, C_2$	a^1_1 . . . . ⊃ $\lfloor C \to A_1, C_1$
	03	α				
	04	α				
	05	α				
	06	α				
	07	α				$\overline{ASK'}+1 \to ASK$
	08	α	a^1_4 . . . . . ⊃ $\lfloor CRY \to A_4$, $\lfloor 0 \to C_4$	a^1_3 . . . . ⊃ $\lfloor CRY \to A_3$, $\lfloor 0 \to C_3$	a^1_2 . . . . ⊃ $\lfloor CRY \to A_2$, $\lfloor 0 \to C_2$	ASK^1_7 . ⊃ $\overline{AK'}+1 \xrightarrow{j} AK$ $ASK^0_7 \cdot ASK^1_2 \cdot ASK^0_1$. ⊃ $\lfloor 10 \to AK$ a^1_1 . . . . ⊃ $\lfloor CRY \to A_1$, $\lfloor 0 \to C_1$
	09	α	$(ASK^1_7 + ASK^0_0) \cdot a^1_4$ ⊃ $\lfloor SHL \to A_4$ a^1_4 . . . . ⊃ $\lfloor SHL \to B_4$ $(A_{4.9} \equiv D_{4.9}) \cdot a^1_4$. ⊃ $\lfloor C \to D_4$	$(ASK^1_7 + ASK^0_0) \cdot a^1_3$ . . . ⊃ $\lfloor SHL \to A_3$ a^1_3 . . . . ⊃ $\lfloor SHL \to B_3$ $[(A_{3.9} \equiv D_{3.9}) \cdot III] + [(A_{4.9} \equiv D_{4.9}) \cdot IV_3]$ ⊃ $\lfloor C \to D_3$	$(ASK^1_7 + ASK^0_0) \cdot a^1_2$ . . . . ⊃ $\lfloor SHL \to A_2$ a^1_2 . . . . ⊃ $\lfloor SHL \to B_2$ $[(A_{2.9} \equiv D_{2.9}) \cdot II_2] + [(A_{4.9} \equiv D_{4.9}) \cdot IV_2]$ ⊃ $\lfloor C \to D_2$	$\overline{AK'}+1 \xrightarrow{j} AK$ $\bar{f}_4$ . . ⊃ $\lfloor 05 \to AK$ f_4 . . ⊃ $\lfloor 06 \to AK$ $ASK^0_7 \cdot ASK^1_1$ . . ⊃ $\lfloor 0 \to AEP$ $(ASK^1_7 + ASK^0_0) \cdot a^1_1$ . . . . ⊃ $\lfloor SHL \to A_1$ a^1_1 . . . . ⊃ $\lfloor SHL \to B_1$ $[(A_{1.9} \equiv D_{1.9}) \cdot I] + [(A_{4.9} \equiv D_{4.9}) \cdot IV_1] + [(A_{2.9} \equiv D_{2.9}) \cdot II_1]$ ⊃ $\lfloor C \to D_1$
		β	$(ASK^1_7 + ASK^0_0 + A^0_{4.9}) \cdot a^1_4$ ⊃ $\lfloor pad \to A_4, C_4$	$(ASK^1_7 + ASK^0_0) \cdot [(A^0_{3.9} \cdot III) + (A^0_{4.9} \cdot IV_3)]$ ⊃ $\lfloor pad \to A_3, C_3$	$(ASK^1_7 + ASK^0_0) \cdot [(A^0_{2.9} \cdot II_2) + (A^0_{4.9} \cdot IV_2)]$ ⊃ $\lfloor pad \to A_2, C_2$	$(ASK^1_7 + ASK^0_0) \cdot [(A^0_{1.9} \cdot I) + (A^0_{4.9} \cdot IV_1) + (A^0_{2.9} \cdot II_1)]$ ⊃ $\lfloor pad \to A_1, C_1$
	10	α	a^1_4 . . . . . ⊃ $A_4 \xleftrightarrow{j} B_4$	a^1_3 . . . . ⊃ $A_3 \xleftrightarrow{j} B_3$	a^1_2 . . . . ⊃ $A_2 \xleftrightarrow{j} B_2$	a^1_1 . . . . ⊃ $A_1 \xleftrightarrow{j} B_1$
	11	α	$Z^0_4 \cdot a^1_4$ . . ⊃ $\lfloor C \to B_4$ $(Z_4 \neq Y_4) \cdot a^1_4$. ⊃ $\lfloor C \to A_4$ $(D_{4.9} \neq Y_4) \cdot a^1_4$. ⊃ $\lfloor C \to D_4$ IV . . . . ⊃ $\lfloor A_{4.9}\ sign \to Z_4$	$(Z^0_3 \cdot III) + (Z^0_4 \cdot IV_3)$ . . ⊃ $\lfloor C \to B_3$ $[(Z_3 \neq Y_3) \cdot III] + [(Z_4 \neq Y_4) \cdot IV_3]$. ⊃ $\lfloor C \to A_3$ $[(D_{3.9} \neq Y_3) \cdot III] + [(D_{4.9} \neq Y_4) \cdot IV_3]$. ⊃ $\lfloor C \to D_3$ III . . . . ⊃ $\lfloor A_{3.9}\ sign \to Z_3$	$(Z^0_2 \cdot II_2) + (Z^0_4 \cdot IV_2)$ . . ⊃ $\lfloor C \to B_2$ $[(Z_2 \neq Y_2) \cdot II_2] + [(Z_4 \neq Y_4) \cdot IV_2]$ ⊃ $\lfloor C \to A_2$ $[(D_{2.9} \neq Y_2) \cdot II_2] + [(D_{4.9} \neq Y_4) \cdot IV_2]$ ⊃ $\lfloor C \to D_2$ II . . . . ⊃ $\lfloor A_{2.9}\ sign \to Z_2$	$\lfloor 0 \to AK_{11-1}$, $\lfloor 1 \to AK_0$ $\overline{AK'}+1 \xrightarrow{j} AK$ $(Z^0_1 \cdot I) + (Z^0_4 \cdot IV_1) + (Z^0_2 \cdot II_1)$ . . ⊃ $\lfloor C \to B_1$ $[(Z_1 \neq Y_1) \cdot I] + [(Z_4 \neq Y_4) \cdot IV_1] + [(Z_2 \neq Y_2) \cdot II_1]$. ⊃ $\lfloor C \to A_1$ $[(D_{1.9} \neq Y_1) \cdot I] + [(D_{4.9} \neq Y_4) \cdot IV_1] + [(D_{2.9} \neq Y_2 \cdot II_1]$ ⊃ $\lfloor C \to D_1$ I . . . . ⊃ $\lfloor A_{1.9}\ sign \to Z_1$

C.A.N 6-21-61

MULTIPLY

OP CODE DESCRIPTION. MUL multiplies the content of A by the content of the selected Memory Element register. The signed product is left in AB by the instruction.

SPECIAL FEATURES. The ASK counter is used to count the number of multiply-step (MS) partial-add (PAD) loop iterations. In this instruction the Z and Y flip-flops are used in the sign control logic. MUL uses the Carry Coupling Units and the Shift Coupling Units.

DETAILS. The configured operand (multiplicand) is loaded into D at $QK^{21\alpha}$. The QK execution logic is identical to that of a LDD, except for the pulses indicated on the MUL time chart. $QK^{14\alpha}$ starts the AK counter, which controls the multiplication logic.

The pulses clearing and presetting the ASK counter occur in $AK^{00\alpha}$ and $AK^{01\alpha}$, respectively. In the example ASK is preset to 170.

At $AK^{01\alpha}$ Z is cleared in the active sign quarters and the content of A (multiplier) is transferred into B. The C register is also cleared as a preliminary to storing the partial carries in the succeeding steps.

In $AK^{02\alpha}$ the multiplicand in D is made positive. Y_i is used to remember the original sign of the multiplicand. The multiplier in B is also made positive by complementing B. Z_i is used to remember the original sign of the multiplier.

The first partial-add pulse is fired off in $AK^{02\beta}$. The pad pulses are always conditional on the right-most bit in B being in the ONE state at the time the pulse is fired off. The first ASK count pulse also occurs at this time.

The MS - PAD loop is now entered. ASK records each traverse of this loop. (The multiply step and partial-add logic are fully explained in Chapter 14.) The MS pulse occurs in $AK^{03\alpha}$ and the PAD pulse in $AK^{03\beta}$. Note that each MS pulse shifts the content of AB one bit to the right.

When ASK reaches the ZERO state (ASK_7^0) AK leaves the loop.

At $AK^{08\alpha}$ a full carry pulse occurs. After this pulse, the magnitude of the product is contained in the AB register. (The right-most bit in B is a duplicate of the sign bits.) If $Z_i \neq Y_i$, i.e., if the sign of the multiplicand and multiplier were not originally the same, AB is complemented, i.e., made negative. The multiplicand is also given its original sign, as remembered by Y_i.

The overflow bits are left cleared.

MUL ILLUSTRATIVE EXAMPLE

QHK	AK	ASK	Y_1	D_1	C_1	Z_1	A_1	B_1	OPERATION
14d START	00d	x x x x x x x	x	x x x x x x x x x	x x x x x x x x x	x	1 0 0 0 1 1 1 0 0	x x x x x x x x x	① OPERAND (MULTIPLICAND) Loaded into D
21d F→D	01d	0 0 0 0 0 0 0	0	0 0 0 0 0 0 0 0 0	x x x x x x x x x	x	1 0 0 0 1 1 1 0 0	x x x x x x x x x	② MULTIPLIER copied from A into B
	02 α	1 1 1 1 0 0 0	1	1 0 0 1 1 1 1 0 1	0 0 0 0 0 0 0 0 0	0	1 0 0 0 1 1 1 0 0	1 0 0 0 1 1 1 0 0	③ MULTIPLIER (B) & MULTIPLICAND (D) made positive numbers; signs remembered in Z & Y
	02 β	1 1 1 1 0 0 1	1	0 1 1 0 0 0 0 1 0	0 0 0 0 0 0 0 0 0	1	0 0 0 0 0 0 0 0 0	0 1 1 1 0 0 0 1 1	PAD ④ PAD loads A with D
	03 α	1 1 1 1 0 0 1	1	0 1 1 0 0 0 0 1 0	0 0 0 0 0 0 0 0 0	1	0 1 1 0 0 0 0 1 0	0 1 1 1 0 0 0 1 1	MS
	03 β	1 1 1 1 0 1 0	1	0 1 1 0 0 0 0 1 0	0 0 0 0 0 0 0 0 0	1	0 0 1 1 0 0 0 0 1	0 0 1 1 1 0 0 0 1	PAD
	03 α	1 1 1 1 0 1 0	1	0 1 1 0 0 0 0 1 0	0 0 1 0 0 0 0 0 0	1	0 1 0 1 0 0 0 1 1	0 1 1 1 0 1 0 0 1	MS
	03 β	1 1 1 1 0 1 1	1	0 1 1 0 0 0 0 1 0	0 0 1 0 0 0 0 0 0	1	0 0 0 0 1 0 0 0 1	1 0 0 1 1 1 0 0 0	—
	03 α	1 1 1 1 0 1 1	1	0 1 1 0 0 0 0 1 0	0 0 1 0 0 0 0 0 0	1	0 0 0 0 1 0 0 0 1	1 0 0 1 1 1 0 0 0	MS
	03 β	1 1 1 1 1 0 0	1	0 1 1 0 0 0 0 1 0	0 0 0 0 0 0 0 0 0	1	0 0 1 0 0 1 0 0 0	1 1 0 0 1 1 1 0 0	—
	03 α	1 1 1 1 1 0 0	1	0 1 1 0 0 0 0 1 0	0 0 0 0 0 0 0 0 0	1	0 0 1 0 0 1 0 0 0	1 1 0 0 1 1 1 0 0	MS
	03 β	1 1 1 1 1 0 1	1	0 1 1 0 0 0 0 1 0	0 0 0 0 0 0 0 0 0	1	0 0 0 1 0 0 1 0 0	0 1 1 0 0 1 1 1 0	—
	03 α	1 1 1 1 1 0 1	1	0 1 1 0 0 0 0 1 0	0 0 0 0 0 0 0 0 0	1	0 0 0 1 0 0 1 0 0	0 1 1 0 0 1 1 1 0	MS
	03 β	1 1 1 1 1 1 0	1	0 1 1 0 0 0 0 1 0	0 0 0 0 0 0 0 0 0	1	0 0 0 0 1 0 0 1 0	0 0 1 1 0 0 1 1 1	PAD
	03 α	1 1 1 1 1 1 0	1	0 1 1 0 0 0 0 1 0	0 0 0 0 0 0 0 1 0	1	0 1 1 0 1 0 0 0 0	0 0 1 1 0 0 1 1 1	MS
	03 β	1 1 1 1 1 1 1	1	0 1 1 0 0 0 0 1 0	0 0 0 0 0 0 0 0 0	1	0 0 1 1 0 1 0 1 0	0 0 0 1 1 0 0 1 1	PAD
	03 α	1 1 1 1 1 1 1	1	0 1 1 0 0 0 0 1 0	0 0 1 0 0 0 0 1 0	1	0 1 0 1 0 1 0 0 0	0 0 0 1 1 0 0 1 1	MS
	03 β	0 0 0 0 0 0 0	1	0 1 1 0 0 0 0 1 0	0 0 1 0 0 0 0 0 0	1	0 0 0 0 1 0 1 1 0	0 0 0 0 1 1 0 0 1	PAD
	03 α	0 0 0 0 0 0 0	1	0 1 1 0 0 0 0 1 0	0 0 1 0 0 0 0 1 0	1	0 1 1 0 1 0 1 0 0	0 0 0 0 1 1 0 0 1	MS
	03 β	0 0 0 0 0 0 1	1	0 1 1 0 0 0 0 1 0	0 0 1 0 0 0 0 1 0	1	0 0 0 1 1 0 1 0 0	0 0 0 0 0 1 1 1 0	—
	06d	0 0 0 0 0 0 1	1	0 1 1 0 0 0 0 1 0	0 0 1 0 0 0 0 1 0	1	0 0 0 1 0 1 0 0 0	0 0 0 0 0 1 1 0 0	① Full carry (CRY) propogated through A
	07d	0 0 0 0 0 0 1	1	0 1 1 0 0 0 0 1 0	0 0 1 0 0 0 0 1 0	1	0 0 0 1 0 1 0 0 0	0 0 0 0 0 1 1 0 0	
	08d	0 0 0 0 0 0 1	1	0 1 1 0 0 0 0 1 0	0 0 1 0 0 0 0 1 0	1	0 0 0 1 0 1 0 0 0	0 0 0 0 0 1 1 0 0	② Initial sign of operand (D) restored. Sign of product in AB determined and affixed
	09d	0 0 0 0 0 0 1	1	0 1 1 0 0 0 0 1 0	0 0 1 0 0 0 0 1 0	1	0 0 0 1 0 1 0 0 0	0 0 0 0 0 1 1 0 0	
	00d	0 0 0 0 0 0 1	1	1 0 0 1 1 1 1 0 1	0 0 1 0 0 0 0 1 0	0	0 1 0 1 0 1 0 0 0	0 0 0 0 0 1 1 0 0	

MS-PAD Loop

① Initial MS always shifts AB one bit to the right

② PAD is conditional on rightmost bit in B being a 1

③ Loop is left when $ASK_7^0 \cdot AK_\alpha^{03}$ occurs.

④ Final ASK value is always 0000001

⑤ The PAD pulse is never fired in the last 03 state, since B_{10} contains the sign bit that was made positive

MULTIPLICATION ILLUSTRATIVE EXAMPLE

OPERAND (Loaded in D from memory) .... | 1 0 0 1 1 1 1 0 1 |

MULTIPLIER (left in A from previous instruction)... | 1 0 0 0 1 1 1 0 0 |

FRACTURE

MULTIPLY (76)

PK 24	α	$1^{00} \rightarrow PK$

00	α	QI^{start} . . . . ⊃	$1^{start} \diamond FK$, $1^{0} \rightarrow PI_1$
01	α		
		SEE QKM TIMING	
09	α		
10	α		$1^{0} \rightarrow E$
11	α		$1^{13} \rightarrow QK$ $D \xrightarrow{1} E$
11	β		$1^{P} \rightarrow E$
QK 13	α		$M \xrightarrow[a.p]{0.1} E$ $1^{0} \rightarrow AK_{11-1}$, $1^{1} \rightarrow AK_0$ $QKIR_{cf3-8} \xrightarrow{j} AKIR_{cf9-8}$ $QKIR^{ext\ acts-1} \xrightarrow{j} AKIR_{cf7-4}$ $QKIR_{op6-1} \xrightarrow{j} AKIR_{op6-1}$
13	β		$1^{P} \rightarrow E$
14	α		$1^{21} \rightarrow QK$ $1^{0}_{SE} \rightarrow E$ $1^{1} \rightarrow AEP$ $1^{0} \rightarrow D$ $1^{start} \diamond AK$
14	β		$1^{C}_{SE} \rightarrow E$
21	α		$E \xrightarrow{1} D$
22	α		
23	α		$M \xrightarrow{0.1} E$ $1^{0} \rightarrow EB$ $1^{31} \rightarrow QK$
31	α		

OP Class Decoders Lines Up: $QKIR^{AK}$, $QKIR^{AESK}$, $QKIR^{load}$, $QKIR^{ld}$ & $QKIR^{D}$

OPERATE (ARITHMETIC ELEMENT: $N^{o}_{2.8} \cdot N'_{2.7}$) OPR^{AE} 04

24	α		
25	α	$QB' + AEB$ . . . . ⊃ AK^{o}_{0} . . . . ⊃ " . . . . ⊃ " . . . . ⊃ (NOTE: $AEB = AK^{o}_{0}$)	$\overline{PK'}+1 \rightarrow PK$ $1^{0} \rightarrow AK_{11-1}$, $1^{1} \rightarrow AK_0$ $N_{2.6-2.1} \xrightarrow{j} AKIR_{op6-1}$ $N_{1.9-1.4} \xrightarrow{j} AKIR_{cf9-4}$
26	α		$1^{31} \rightarrow PK$ $1^{1} \rightarrow AEP$ $1^{start} \diamond AK$
31	α	$PI^{ch\ seq}$ . . . . ⊃	$1^{1} \rightarrow PI_3$

OP Class Decoder Lines Up: $PKIR^{def}$, $PKIR^{dis}$, $PKIR^{AE}$

00	α				$1^{start} \diamond AK$ . . . ⊃ $\overline{AK'}+1 \rightarrow AK$ $1^{start} \diamond AK$ . . . ⊃ $1^{0} \rightarrow ASK$
01	α	IV . . . . ⊃ $1^{0} \rightarrow Z_4$ a'_4 . . . . ⊃ $A_4 \xrightarrow{j} B_4$ a'_4 . . . . ⊃ $1^{0} \rightarrow C_4$	III . . . . ⊃ $1^{0} \rightarrow Z_3$ a'_3 . . . . ⊃ $A_3 \xrightarrow{j} B_3$ a'_3 . . . . ⊃ $1^{0} \rightarrow C_3$	II . . . . ⊃ $1^{0} \rightarrow Z_2$ a'_2 . . . . ⊃ $A_2 \xrightarrow{j} B_2$ a'_2 . . . . ⊃ $1^{0} \rightarrow C_2$	$1^{preset} \rightarrow ASK$ I . . . . ⊃ $1^{0} \rightarrow Z_1$ a'_1 . . . . ⊃ $A_1 \xrightarrow{j} B_1$ a'_1 . . . . ⊃ $1^{0} \rightarrow C_1$
02	α	IV . . . . ⊃ $1^{A_{4.9}\ sign} \rightarrow Z_4$ a'_4 . . . . ⊃ $1^{0} \rightarrow A_4$ $(B'_{4.9} \cdot a'_4)$ . . . . . ⊃ $1^{C} \rightarrow B_4$ $(Y'_4 \cdot a'_4)$ . . . . . ⊃ $1^{C} \rightarrow D_4$	III . . . . ⊃ $1^{A_{3.9}\ sign} \rightarrow Z_3$ a'_3 . . . . ⊃ $1^{0} \rightarrow A_3$ $(B'_{3.9} \cdot III) + (B'_{4.9} \cdot IV_3)$ ⊃ $1^{C} \rightarrow B_3$ $(Y'_3 \cdot III) + (Y'_4 \cdot IV_3)$ ⊃ $1^{C} \rightarrow D_3$	II . . . . ⊃ $1^{A_{2.9}\ sign} \rightarrow Z_2$ a'_2 . . . . ⊃ $1^{0} \rightarrow A_2$ $(B'_{2.9} \cdot II_2) + (B'_{4.9} \cdot IV_2)$ . . ⊃ $1^{C} \rightarrow B_2$ $(Y'_2 \cdot II_2) + (Y'_4 \cdot IV_2)$ . . ⊃ $1^{C} \rightarrow D_2$	$\overline{ASK'}+1 \rightarrow ASK$ I . . . . ⊃ $1^{A_{1.9}\ sign} \rightarrow Z_1$ a'_1 . . . . ⊃ $1^{0} \rightarrow A_1$ $(B'_{1.9} \cdot I)+(B'_{4.9} \cdot IV_1)+(B'_{2.9} \cdot II_1)$ ⊃ $1^{C} \rightarrow B_1$ $(Y'_1 \cdot I)+(Y'_4 \cdot IV_1)+(Y'_2 \cdot II_1)$ ⊃ $1^{C} \rightarrow D_1$
	β	$a'_4 \cdot [(B'_{4.1} \cdot f_4)+(B'_{3.1} \cdot f_2)+(B'_{2.1} \cdot f_3)+(B'_{1.1} \cdot f_1)]$ ⊃ $1^{pad} \rightarrow A_4, C_4$	$a'_3 \cdot \{[B'_{3.1} \cdot (f_2+f_4)]+[B'_{2.1} \cdot f_3]+[B'_{1.1} \cdot f_1]\}$ ⊃ $1^{pad} \rightarrow A_3, C_3$	$a'_2 \cdot \{[B'_{2.1} \cdot (f_3+f_4)] + [B'_{1.1} \cdot (f_1+f_2)]\}$. ⊃ $1^{pad} \rightarrow A_2, C_2$	$(a'_1 \cdot B'_{1.1})$ . . . . . . ⊃ $1^{pad} \rightarrow A_1, C_1$
AK 03	α	a'_4 . . . . ⊃ $1^{mult\ step} \rightarrow A_4, C_4$ a'_4 . . . . ⊃ $1^{SHR} \rightarrow B_4$	IV_3 . . . . . ⊃ $1^{mult\ step\ sign} \rightarrow A_{3.9}, C_{3.9}$ a'_3 . . . . ⊃ $1^{mult\ step} \rightarrow A_3, C_3$ a'_3 . . . . ⊃ $1^{SHR} \rightarrow B_3$	IV_2 . . . . . . ⊃ $1^{mult\ step\ sign} \rightarrow A_{2.9}, C_{2.9}$ a'_2 . . . . ⊃ $1^{mult\ step} \rightarrow A_2, C_2$ a'_2 . . . . ⊃ $1^{SHR} \rightarrow B_2$	$\overline{ASK'}+1 \rightarrow ASK$ $\overline{AK'}+1 \rightarrow AK$ ASK^{o}_{7} . . . . ⊃ $1^{0} \rightarrow AEP$ $ASK^{o}_{7} \cdot \bar{f}_4$ . . . ⊃ $1^{05} \rightarrow AK$ $ASK^{o}_{7} \cdot f_4$ . . . ⊃ $1^{06} \rightarrow AK$ $(II_1 + IV_1)$ . . . . ⊃ $1^{mult\ step\ sign} \rightarrow A_{1.9}, C_{1.9}$ a'_1 . . . . ⊃ $1^{mult\ step} \rightarrow A_1, C_1$ a'_1 . . . . ⊃ $1^{SHR} \rightarrow B_1$
	β	$a'_4 \cdot [(B'_{4.1} \cdot f_4)+(B'_{3.1} \cdot f_2)+(B'_{2.1} \cdot f_3)+(B'_{1.1} \cdot f_1)]$ ⊃ $1^{pad} \rightarrow A_4, C_4$	$a'_3 \cdot \{[B'_{3.1} \cdot (f_2+f_4)]+[B'_{2.1} \cdot f_3]+[B'_{1.1} \cdot f_1]\}$ ⊃ $1^{pad} \rightarrow A_3, C_3$	$a'_2 \cdot \{[B'_{2.1} \cdot (f_3+f_4)] + [B'_{1.1} \cdot (f_1+f_2)]\}$. ⊃ $1^{pad} \rightarrow A_2, C_2$	$(a'_1 \cdot B'_{1.1})$ . . . . . . ⊃ $1^{pad} \rightarrow A_1, C_1$
04	α				
05	α				
06	α				
07	α				
08	α	a'_4 . . . . ⊃ $1^{0} \rightarrow C_4$ $a'_4 \cdot \{\overline{A'_4}+(\overline{A'_3} \cdot f_4)+[\overline{A'_2} \cdot (f_1+f_3)]+(\overline{A'_1} \cdot f_1)\}$ ⊃ $1^{CRY} \rightarrow C_4$	a'_3 . . . . ⊃ $1^{0} \rightarrow C_3$ $a'_3 \cdot \{\overline{A'_3}+(\overline{A'_4} \cdot f_4)+[\overline{A'_2} \cdot (f_1+f_3)]+(\overline{A'_1} \cdot f_1)\}$ ⊃ $1^{CRY} \rightarrow A_3$	a'_2 . . . . ⊃ $1^{0} \rightarrow C_2$ $a'_2 \cdot \{\overline{A'_2}+[\overline{A'_4} \cdot (f_1+f_3)]+[\overline{A'_3} \cdot (f_1+f_3)]+[\overline{A'_1} \cdot (f_1+f_2)]\}$ ⊃ $1^{CRY} \rightarrow A_2$	a'_1 ⊃ $1^{0} \rightarrow C_1$ $a'_1 \cdot \{\overline{A'_1}+(\overline{A'_4} \cdot f_1)+(\overline{A'_3} \cdot f_1)+[\overline{A'_2} \cdot (f_1+f_2)]\}$ ⊃ $1^{CRY} \rightarrow A_1$
09	α	IV . . . . ⊃ $1^{0} \rightarrow Z_4$ $(Y_4 \neq Z_4) \cdot a'_4$ . . . ⊃ $1^{C} \rightarrow A_4$ (") " . . . . ⊃ $1^{C} \rightarrow B_4$ $(Y'_4 \cdot a'_4)$ . . . . . ⊃ $1^{C} \rightarrow D_4$	III . . . . ⊃ $1^{0} \rightarrow Z_3$ $[(Y_3 \neq Z_3) \cdot III]+[(Y_4 \neq Z_4) \cdot IV_3]$ ⊃ $1^{C} \rightarrow A_3$ ["]+["] ⊃ $1^{C} \rightarrow B_3$ $(Y'_3 \cdot III) + (Y'_4 \cdot IV_3)$ . . ⊃ $1^{C} \rightarrow D_3$	II . . . . ⊃ $1^{0} \rightarrow Z_2$ $[(Y_2 \neq Z_2) \cdot II_2] + [(Y_4 \neq Z_4) \cdot IV_2]$. ⊃ $1^{C} \rightarrow A_2$ ["] + ["] . ⊃ $1^{C} \rightarrow B_2$ $(Y'_2 \cdot II_2) + (Y'_4 \cdot IV_2)$ . . ⊃ $1^{C} \rightarrow D_2$	$1^{0} \rightarrow AK_{11-1}$, $1^{1} \rightarrow AK_0$ $\overline{AK'}+1 \rightarrow AK$ I . . . . ⊃ $1^{0} \rightarrow Z_1$ $[(Y_1 \neq Z_1) \cdot I]+[(Y_4 \neq Z_4) \cdot IV_1]+[(Y_2 \neq Z_2) \cdot II_1]$ ⊃ $1^{C} \rightarrow A_1$ ["]+["]+["] ⊃ $1^{C} \rightarrow B_1$ $(Y'_1 \cdot I)+(Y'_4 \cdot IV_1)+(Y'_2 \cdot II_1)$. ⊃ $1^{C} \rightarrow D_1$

C.A.N. 6-22-61